dlkit Documentation

Release 0.0.2

Jeff Merriman

May 31, 2017

Contents

This documentation for the Digital Learning Toolkit (DLKit), like the toolkit itself, is still under development. It currently covers only a small handful of the 180 service packages and over 10,000 educational service APIs that have been defined by MIT's Office of Digital Learning and its collaborators to date.

The DLKit codebase is generated from the Open Service Interface Definitions (OSIDs), an extensive and growing suite of interface contract specifications that describe the integration points among the core services and components that make up modern educational systems.

Note that this documentation is intended for API consumers. However, at the heart of DLKit is an integration stack that is even more closely aligned with the OSID specifications. This has been designed to allow third parties to extend the library with alternative or additional implementations of any of the defined services for the purposes of service integration, technology migration, service adaptation, etc. Documentation written for service implementers and system integrators, including implementation notes and compliance information, will be provided elsewhere.

The complete OSID specification can be found at http://osid.org/specifications.

Currently, DLKit works with Python 2.7, 3.4, 3.5, and 3.6.

If you are interested in learning more about the DLKit Python libraries documented here, please contact dlkit-info@mit.edu

CHAPTER 1

Get DLKit

The latest release of DLKit and related dependencies are available here: https://bitbucket.org/cjshaw/

The core MongoDB implemenation is located in the https://bitbucket.org/cjshaw/dlkit repository, which has sub-module dependencies on a number of the other repositories on the site. More information about this can be found in the introduction section.

CHAPTER 2

Testing

Unit tests and limited functional tests for DLKit are available here: https://bitbucket.org/cjshaw/dlkit-tests

CHAPTER 3

Contents:

Introduction to DLKit

Basic OSID terms

Some familiarity with the OSID models and terms will help understand how to use DLKit.

Each service has its own set of packages. Some commonly used DLKit services include `assessment`, `learning`, and `repository`. Within each service, users encounter three types of objects: `managers`, `sessions`, and `objects`. `sessions` are the verbs of the system (what you want to do), while `objects` are the nouns (what you are working on). For example, to look up learning objectives in the `learning` service, you would use a `ObjectiveLookupSession`, which has all the methods you need to look up objectives, like `get_objective()`, `get_objectives_by_ids()`, etc.

Managers

Managers act as the entry points to each service by granting access to sessions. For basic DLKit usage, users will most likely use the service managers to get the service catalog, which has been overloaded to include the most used methods (i.e. the catalog for `learning` is an `ObjectiveBank`).

Advanced users can use the service manager in a strict OSID fashion as well.

Sessions

Sessions allow users to perform a certain action on a desired `object`. For basic users, they will most likely never encounter a `session`, so the usage is only address in the advanced usage page.

Objects

In each OSID service, there are typically two types of `objects` – the overall catalog, as well as the more granular object that resides in the catalogs.

Catalogs are generally used for authorization (who can do what actions to what objects), and in DLKit these authorizations flow down. If catalog B is a parent of catalog C, and I can create things in catalog B, then I can automatically create things in catalog C as well.

For example, in the `learning` service, the catalog is an `ObjectiveBank`, and the more granular resources are `objectives`.

Usage

While DLKit is an implementation of the OSIDs, a convenience layer has been built on top of the OSID implementation. This layer is called `dlkit.services`, and it is recommended that new developers start working with this layer, first. General instructions for these are outlined in the *Basic DLKit Usage* section.

If users find certain functionality missing or inconvenient, then they should check out the *Advanced DLKit Usage* section.

Directory Structure

Until we have a good way to package DLKit and make it available as a Python installable, the easiest way to use it is to `git clone` it into your project directory along with its submodules. An example directory structure is:

```
my_project/
    |—-__init__.py
    |—-my_project/
    |—-dlkit/
        |—-aws_adapter/
        |—-handcar/
        |—-mongo/records/
        |—-primordium/
    |—-dlkit_runtime/
        |—-configs.py
        |—-registry.py
```

Dependencies

DLKit has several dependencies that are also listed as git submodules. They are relisted here for convenience, along with their git repositories:

- AWS Adapter (for storing / retrieving files from Amazon AWS S3): https://bitbucket.org/cjshaw/aws_adapter
- Handcar (MC3 learning service-based implementation): https://bitbucket.org/cjshaw/handcar
- Primordium (basic object types): https://bitbucket.org/cjshaw/primordium
- Record extensions (for extending objects in the MongoDB implementation): https://bitbucket.org/cjshaw/records

Basic DLKit Usage

Runtimes and basic DLKit usage are also covered in the sample *Tutorial: DLKit Learning Service Basics*. The information is discussed here in a more general sense. The examples below explicitly use the Django runtime, but you can modify them to suit your custom runtime.

Configuring Runtimes

To access `managers` using a runtime, you first need to create a user proxy, which wraps the user object and allows DLKit to calculate authorizations. You then pass the proxy into the runtime and request managers.:

```
from dlkit_django import RUNTIME, PROXY_SESSION

condition = PROXY_SESSION.get_proxy_condition()
condition.set_http_request(request)
proxy = PROXY_SESSION.get_proxy(condition)

lm = RUNTIME.get_service_manager('LEARNING', proxy)
```

Service Managers and Catalogs

In the `services` convenience layer, you typically deal with two objects for each service, the `manager` and a `catalog`, which subclass a subset of OSID classes. This means that you do not have to worry about managing OSID sessions – this is managed for you and simplifies your interactions with DLKit.

Service Managers

Service managers typically give you access to all methods that do not require a specific catalog. For example, the `LearningManager` lets you create, update, query, and delete `ObjectiveBanks` in the `learning` service.:

```
bank = lm.get_objective_bank(bank_id)
```

Service Catalogs

The catalogs you get back from a `service manager` typically give you access too all objects within that catalog. So an `ObjectiveBank` lets you create, update, query, and delete learning objectives and activities.:

```
objectives = bank.get_objectives()
```

Creating and Updating objects

To create or update an object, DLKit uses forms.:

```
form = lm.get_objective_bank_form_for_create([])
form.display_name = "My new test bank"
new_bank = lm.create_objective_bank(form)
```

The optional parameter (on creation) allows you to specify which record extensions, if any, you want applied to the object. For example, an Open edX-type objective bank might have additional methods that allow you to export content in OLX format.

Once you have an object's `id`, you can also update it with a form.:

```
form = lm.get_objective_bank_form_for_update(bank_id)
form.description = "For testing with"
updated_bank = lm.update_objective_bank(form)
```

To update objects within a catalog, you would do the same thing, but via a `catalog` object.:

```
form = bank.get_objective_form_for_create([])
form = bank.get_objective_form_for_update(objective_id)
```

Advanced DLKit Usage

Sometimes you may need to use DLKit in a stricter OSID fashion, bypassing the `services` convenience methods discussed in *Basic DLKit Usage*. To use DLKit in a more OSID-y way, you would manually manage the various admin / lookup / etc. sessions, from either a service manager or a service catalog.

Currently, the main use case when you would need to manage sessions yourself is if you want to look up objects without knowing their catalog IDs. For example, you have an `objective_id` but not the `objective_bank_id` and need to grab the objective. In the future, we plan to build that capability into the `services` convenience layer, to reduce the need for manual session management.:

```
ols = lm.get_objective_lookup_session()
objective = ols.get_objective(objective_id)
```

Configuring DLKit

DLKit requires the use of a runtime environment, from which you configure the user proxy object, get managers, and define which implementations are used for each service (among other things). For example, you might use the built-in MongoDB `learning` service, or you may decide to use the MC3 RESTful service, Handcar.

Two examples are available, one for Open edX and another for Django. You can also build your own, using these as templates:

- Django runtime: https://bitbucket.org/cjshaw/dlkit_django
- Open edX runtime: https://bitbucket.org/cjshaw/dlkit_edx

Both runtimes require the addition of `configs.py` and `registry.py` files. These allow different instances of DLKit to behave differently, depending on the project's needs.

NOTE: With the Django runtime, you can establish the configuration values in your project's `settings.py` file, which is then referenced in the runtime files below. This is for your convenience. You can also configure the settings directly in the files below, as shown in the Open edX runtime.

configs.py

This file allows you to define various parameters used in the production runtime as well as a testing runtime. The only required values are for `BOOTSTRAP` and `SERVICE` (and for running the DLKit tests, `TEST_SERVICE`).

`SERVICE` should point to the implementation you want to use for each service (i.e. Handcar for the `learning` service, MongoDB for the assessment service).

Currently, this file also lets you manage the following

- Amazon AWS credentials, if using the AWS shim adapter.
- MongoDB authority, for identifying objects / IDs created by this instance.
- MongoDB database prefix, to namespace your collections and documents.
- MongoDB fields to index on.

- MongoDB fields to search on, for keyword searches.
- MongoDB host URI, if using a sharded repository.

An example skeleton is included in the runtime repository.

registry.py

Similar to `setup.py`, this file includes the various entry points for each service and implementation. For example, if you expect to use the `grading` service from the MongoDB implementation, you need to make sure it has an entry in this file. An example skeleton is included in the repository.

handcar_configs.py

If you are using the *MC3 Handcar* learning service, then you will also need to add in proxy agent keys for the appropriate service / server combination in this file, if you want more than read-access.

Tutorial: DLKit Learning Service Basics

This tutorial is under development. It currently focuses on aspects of the `Learning` service. At the time of this writing, MIT's Office of Digital Learning is supporting a production learning objective management service called the MIT Core Concept Catalog (MC3). DLKit includes an underlying implementation that uses MC3 for persistence. As a result, this tutorial uses examples primarily from this particular service, which deals with managing learning objectives, learning paths and relationships between learning objectives and educational assets, assessments, etc, since there is data available for testing.

All of the other DLKit Interface Specifications build on most of the same patterns outlined in this tutorial, beginning with loading managers. Make sure that the dlkit package is in your python path or install the library.

The Runtime Manager and Proxy Authentication

Service managers are instantiated through a Runtime Manger, which are designed to work with certain runtime environments, like Django or edX/XBlock runtimes. To get access to these runtime environments please contact dlkit-info@mit.edu. Install the runtime environment you want to use and make sure that your Django project's settings.py includes `dlkit_django` or `dlkit_xblock` as appropriate.

Now you can get the `RuntimeManager` root instance for your runtime environment. Note that there is only one, and it gets instantiated at environment launch time, it is thread-safe and used by all consumer application sessions:

```
from dlkit_django import RUNTIME
```

This `runtime` object is your gateway to access all the underlying service managers and their respective service sessions and functionality

The django runtime knows about Django's own services for users. You will have access to an HTTPRequest object that includes an user authentication (the request variable in the examples below). This needs to be encapsulated in a Proxy object:

```
from dlkit_django import PROXY_SESSION
condition = PROXY_SESSION.get_proxy_condition()
condition.set_http_request(request)
proxy = PROXY_SESSION.get_proxy(condition)
```

Or, if you are standing up dlkit in edX, get an XBlockUser() object from the xblock runtime. That object is assumed to be stored the 'xblock_user' variable below:

```
from dlkit_xblock import PROXY_SESSION
condition = PROXY_SESSION.get_proxy_condition()
condition.set_xblock_user(xblock_user)
proxy = PROXY_SESSION.get_proxy(condition)
```

Now you have a Proxy object that holds the user data and eventually other stuff, like locale information, etc, that can be used to instantiate new service Managers, which you can also insert into your HttpRequest.session:

```
from dlkit_django import RUNTIME
request.session.lm = RUNTIME.get_service_manager('LEARNING', proxy)
```

For the duration of the session you can use this for all the other things. that you normally do.

There is a lot more you can do with the `RuntimeManager`, but getting service managers will be the most common task. One of the other important tasks of this manager, is configuration the underlying service stack based on the `configs.py` file and associated helpers. We will cover this later.

Loading the Learning Manager

All consumer applications wishing to use the DLKit Learning service should start by instantiating a `LearningManager` (don't worry about the proxy for now):

```
lm = runtime.get_service_manager('LEARNING')
```

Everything you need to do within the learning service can now be accessed through this manager. An OSID service `Manager` is used like a factory, providing all the other objects necessary for using the service. You should never try to instantiate any other OSID object directly, even if you know where its class definition resides.

The simplest thing you can do with a manager is to inspect its `display_name` and `description` methods. Note that DLKit always returns user-readable strings as `DisplayText` objects. The actual text is available via the `get_text()` method. Other `DisplayText` methods return the `LanguageType`, `ScriptType` and `FormatType` of the text to be displayed:

```
print "Learning Manager successfully instantiated:"
print "   " + lm.get_display_name().get_text()
print "   " + lm.get_description().get_text()
print ("   (this description was written using the " +
    lm.get_description().get_language_type().get_display_label().get_text() +
    " language)\n")
```

Results in something that looks like this:

```
Learning Manager successfully instantiated:
  MC3 Learning Service
  OSID learning service implementation of the MIT Core Concept Catalog (MC3)
  (this description was written using the English language)

  # Note that the implementation name and description may be different for you.
  # It will depend on which underlying service implementation your dlkit library is
  # configured to point to.  More on this later
```

Alternatively, the Python OSID service interfaces also specify property attributes for all basic "getter" methods, so the above could also be written more Pythonically as:

```
print "Learning Manager successfully instantiated:"
print "  " + lm.display_name.text
print "  " + lm.description.text
print ("  (this description was written using the " +
    lm.description.language_type.display_label.text + " language)\n")
```

For the remainder of this tutorial we will use the property attributes wherever available.

Looking up Objective Banks

Managers encapsulate service profile information, allowing a consumer application to ask questions about which functions are supported in the underlying service implementations that it manages:

```
if lm.supports_objective_bank_lookup():
    print "This Learning Manager can be used to lookup ObjectiveBanks"
else:
    print "What a lame Learning Manager. It can't even lookup ObjectiveBanks"
```

The `LearningManager` also provides methods for getting `ObjectiveBanks`. One of the most useful is get_objective_banks(), which will return an iterator containing all the banks known to the underlying implementations. This is also available as a property, so treating `objective_banks` as an attribute works here too:

```
if lm.supports_objective_bank_lookup():
    banks = lm.objective_banks
    for bank in banks:
        print bank.display_name.text
else:
    print "Objective bank lookup is not supported."
```

This will print a list of the names of all the banks, which can be thought of as catalogs for organizing learning objectives and other related information. At the time of this writing the following resulted:

```
Crosslinks
Chemistry Bridge
i2.002
Python Test Sandbox
x.xxx
```

Note that the OSIDs specify to first ask whether a functional area is supported before trying to use it. However, if you wish to adhere to the Pythonic EAFP (easier to ask forgiveness than permission) programming style, managers will throw an `Unimplemented` exception if support is not available:

```
try:
    banks = lm.objective_banks
except Unimplemented:
    print "Objective bank lookup is not supported."
else:
    for bank in banks:
        print bank.display_name.text
```

The object returned from the call to `get_objective_banks()` is an `OsidList` object, which as you can see from the example is just a Python iterator. Like all iterators it is "wasting", meaning that, unlike a Python `list` it will be completely used up and empty after all the elements have been retrieved.

Like any iterator an `OsidList` object can be cast as a more persistent Python list, like so:

```
banks = list(obls.objective_banks)
```

Which is useful if the consuming application needs to keep it around for a while. However, when we start dealing with `OsidLists` from service implementations which may return very large result sets, or where the underlying data changes often, casting as a `list` may not be wise. Developers are encouraged to treat these as iterators to the extent possible, and refresh from the session as necessary.

You can also inspect the number of available elements in the expected way:

```
len(obls.objective_banks)
    # or
banks = obls.objective_banks
len(banks)
```

And walk through the list one-at-a-time, in `for` statements, or by calling `next()`:

```
banks = lm.objective_banks
crosslinks_bank = banks.next() # At the time of this writing, Crosslinks was first
chem_bridge_bank = banks.next() # and Chemistry Bridge was second
```

OSID Ids

To begin working with OSID *objects*, like `ObjectiveBanks` it is important to understand how the OSIDs deal with identity. When an OSID object is asked for its id an OSID `Id` object is returned. This is *not a ‘‘string‘‘*. It is the unique identifier object for the OSID object. Any requests for getting a specific object by its unique identifier will be accomplished through passing this `Id` object back through the service.

`Ids` are obtained by calling an OSID object's `get_id()` method, which also provides an `ident` attribute property associated with it for convenience (`id` is a reserved word in Python so it is not used)

`OsidObject.ident`	Gets the Id associated with this instance of this OSID object.

So we can try this out:

```
crosslinks_bank_id = crosslinks_bank.ident
chem_bridge_bank_id = chem_bridge_bank.ident
```

`Ids` can be compared for equality:

```
crosslinks_bank_id == chem_bridge_bank_id
    # should return False

crosslinks_bank_id in [crosslinks_bank_id, chem_bridge_bank_id]
    # should return True
```

If a consumer wishes to persist the identifier then it should serialize the returned `Id` object, and all Ids can provide a string representation for this purpose:

id_str_to_perist = str(crosslinks_bank_id)

A consumer application can also stand up an Id from a persisted string. There is an implementation of the Id primitive object available through the runtime environment for this purpose. For instance, from the dlkit_django package:

```
from dlkit_django.primordium import Id
crosslinks_bank_id = Id(id_str_to_persist)
```

Once an application has its hands on an `Id` object it can go ahead and retrieve the corresponding Osid Object through a Lookup Session:

```
crosslinks_bank_redux = obls.get_objective_bank(crosslinks_bank_id)
```

We now have two *different* objects representing the *same* Crosslinks bank, which can be determined by comparing `Ids`:

```
crosslinks_bank_redux == crosslinks_bank
    # should be False

crosslinks_bank_redux.ident == crosslinks_bank_id
    # should be True
```

Looking up Objectives

ObjectiveBanks provide methods for looking up and retrieving learning `Objectives`, in bulk, by `Id`, or by `Type`. Some of the more useful methods include:

`ObjectiveBank.can_lookup_objectives`()	Tests if this user can perform `Objective` lookups.
`ObjectiveBank.objectives`	Gets all `Objectives`.
`ObjectiveBank.get_objective`(objective_id)	Gets the `Objective` specified by its `Id`.
`ObjectiveBank.get_objectives_by_genus_type`(...)	Gets an `ObjectiveList` corresponding to the given objective genus `Type` which does not include objectives of genus types derived from the specified `Type`.

So let's try to find an `Objective` in the Crosslinks bank with a display name of "Definite integral". (Note, that there are also methods for querying `Objectives` by various attributes. More on that later.):

```
for objective in crosslinks_bank:
    if objective.display_name.text == 'Definite integral':
        def_int_obj = objective
```

Now we have our hands on an honest-to-goodness learning objective as defined by an honest-to-goodness professor at MIT!

Authorization Hints

Many service implementations will require *authentication* and *authorization* for security purposes *(authn/authz)*. Authorization checks will be done when the consuming application actually tries to invoke a method for which authz is required, and if its found that the currently logged-in user is not authorized, then the implementation will raise a `PermissionDenied` error.

However, sometimes its nice to be able to check in advance whether or not the user is likely to be denied access. This way a consuming application can decide, for instance, to hide or "gray out" UI widgets for doing un-permitted functions. This is what the methods like `can_lookup_objectives` are for. They simply return a `boolean`.

The Objective Object

Objectives inherit from OsidObjects (ObjectiveBanks do too, by the way), which means there are a few methods they share with all other OsidObjects defined by the specification

OsidObject.display_name	Gets the preferred display name associated with this instance of this OSID object appropriate for display to the user.
OsidObject.description	Gets the description associated with this instance of this OSID object.
OsidObject.genus_type	Gets the genus type of this object.

The display_name and description attributes work exactly like they did for ObjectiveBanks and both return a Displaytext object that can be interrogated for its text or the format, language, script of the text to be displayed. We'll get to genus_type in a little bit

Additionally Objectives objects can hold some information particular to the kind of data that they manage:

Objective.has_assessment()	Tests if an assessment is associated with this objective.
Objective.assessment	Gets the assessment associated with this learning objective.
Objective.assessment_id	Gets the assessment Id associated with this learning objective.
Objective.has_cognitive_process()	Tests if this objective has a cognitive process type.
Objective.cognitive_process	Gets the grade associated with the cognitive process.
Objective.cognitive_process_id	Gets the grade Id associated with the cognitive process.
Objective.has_knowledge_category()	Tests if this objective has a knowledge dimension.
Objective.knowledge_category	Gets the grade associated with the knowledge dimension.
Objective.knowledge_category_id	Gets the grade Id associated with the knowledge dimension.

OSID Types

The OSID specification defines Types as a way to indicate additional agreements between service consumers and service providers. A Type is similar to an Id but includes other data for display and organization:

Type.display_name	Gets the full display name of this Type.
Type.display_label	Gets the shorter display label for this Type.
Type.description	Gets a description of this Type.
Type.domain	Gets the domain.

Types also include identification elements so as to uniquely identify one Type from another:

Type.authority	Gets the authority of this Type.
Type.namespace	Gets the namespace of the identifier.
Type.identifier	Gets the identifier of this Type.

Consuming applications will often need to persist Types for future use. Persisting a type reference requires persisting all three of these identification elements.

For instance the MC3 service implementation supports two different types of Objectives, which help differentiate between *topic* type objectives and *learning outcome* type objectives. One way to store, say, the topic type for future programmatic reference might be to put it in a dict:

```
OBJECTIVE_TOPIC_TYPE = {
    'authority': 'MIT-OEIT',
    'namespace': 'mc3-objective',
    'identifier': 'mc3.learning.topic'
    }
```

The OSIDs also specify functions for looking up types that are defined and/or supported, and as one might imagine, there is `TypeLookupSession` specifically designed for this purpose. This session, however, is not defined in the *learning* service package, it is found in the *type* package, which therefore requires a `TypeManager` be instantiated:

```
tm = runtime.get_service_manager('LEARNING', <proxy>)
...
if tm.supports_type_lookup():
    tls = tm.get_type_lookup_session()
```

The `TypeLookupSession` provides a number of ways to get types, two of which are sufficient to get started:

`TypeLookupSession.types`	Gets all the known Types.
`TypeLookupSession.get_type`(namespace, ...)	Gets a `Type` by its string representation which is a combination of the authority and identifier.

For kicks, let's print a list of all the `Types` supported by the implementation:

```
for typ in tls.types:
    print typ.display_label.text

# For the MC3 implementation this should result in a very long list
```

Also, we can pass the `dict` we created earlier to the session to get the actual `Type` object representing the three identification elements we persisted:

```
topic_type = tls.get_type(**OBJECTIVE_TOPIC_TYPE)
print topic_type.display_label.text + ': ' + topic_type.description.text

# This should print the string 'Topic: Objective that represents a learning topic'
```

(More to come)

Assessment

Summary

Assessment Open Service Interface Definitions assessment version 3.0.0

The Assessment OSID provides the means to create, access, and take assessments. An `Assessment` may represent a quiz, survey, or other evaluation that includes assessment `Items`. The OSID defines methods to describe the flow of control and the relationships among the objects. Assessment `Items` are extensible objects to capture various types of questions, such as a multiple choice or an asset submission.

The Assessment service can br broken down into several distinct services:

- Assessment Taking

- Assessment and Item authoring
- Accessing and managing banks of assessments and items

Each of these service areas are covered by different session and object interfaces. The object interfaces describe both the structure of the assessment and follow an assessment management workflow of first defining assessment items and then authoring assessments based on those items. They are:

- `Item` : a question and answer pair
- `Response:` a response to an `Item` question
- `Assessment` : a set of `Items`
- `AssessmentSection:` A grouped set of `Items`
- `AssessmentOffering:` An `Assessment` available for taking
- `AssessmentTaken:` An `AssessmentOffering` that has been completed or in progress

Taking Assessments

The `AssessmentSession` is used to take an assessment. It captures various ways an assessment can be taken which may include time constraints, the ability to suspend and resume, and the availability of an answer.

Taking an `Assessment` involves first navigating through `AssessmentSections`. An `AssessmentSection` is an advanced authoring construct used to both visually divide an `Assessment` and impose additional constraints. Basic assessments are assumed to always have one `AssessmentSection` even if not explicitly created.

Authoring

A basic authoring session is available in this package to map `Items` to `Assessments`. More sophisticated authoring using `AssessmentParts` and sequencing is available in the Assessment Authoring OSID.

Bank Cataloging

`Assessments`, `AssessmentsOffered`, `AssessmentsTaken`, and `Items` may be organized into federateable catalogs called `Banks` .

Sub Packages

The Assessment OSID includes an Assessment Authoring OSID for more advanced authoring and sequencing options. Assessment Open Service Interface Definitions assessment version 3.0.0

The Assessment OSID provides the means to create, access, and take assessments. An `Assessment` may represent a quiz, survey, or other evaluation that includes assessment `Items`. The OSID defines methods to describe the flow of control and the relationships among the objects. Assessment `Items` are extensible objects to capture various types of questions, such as a multiple choice or an asset submission.

The Assessment service can br broken down into several distinct services:

- Assessment Taking
- Assessment and Item authoring
- Accessing and managing banks of assessments and items

Each of these service areas are covered by different session and object interfaces. The object interfaces describe both the structure of the assessment and follow an assessment management workflow of first defining assessment items and then authoring assessments based on those items. They are:

- `Item` : a question and answer pair
- `Response:` a response to an `Item` question
- `Assessment` : a set of `Items`

- `AssessmentSection`: A grouped set of `Items`
- `AssessmentOffering`: An `Assessment` available for taking
- `AssessmentTaken`: An `AssessmentOffering` that has been completed or in progress

Taking Assessments

The `AssessmentSession` is used to take an assessment. It captures various ways an assessment can be taken which may include time constraints, the ability to suspend and resume, and the availability of an answer.

Taking an `Assessment` involves first navigating through `AssessmentSections`. An `AssessmentSection` is an advanced authoring construct used to both visually divide an `Assessment` and impose additional constraints. Basic assessments are assumed to always have one `AssessmentSection` even if not explicitly created.

Authoring

A basic authoring session is available in this package to map `Items` to `Assessments`. More sophisticated authoring using `AssessmentParts` and sequencing is available in the Assessment Authoring OSID.

Bank Cataloging

`Assessments`, `AssessmentsOffered`, `AssessmentsTaken`, and `Items` may be organized into federateable catalogs called `Banks` .

Sub Packages

The Assessment OSID includes an Assessment Authoring OSID for more advanced authoring and sequencing options.

Service Managers

Assessment Profile

class `dlkit.services.assessment.`**`AssessmentProfile`**

Bases: `dlkit.osid.managers.OsidProfile`

The `AssessmentProfile` describes the interoperability among assessment services.

`supports_assessment``()`

Tests for the availability of a assessment service which is the service for taking and examining assessments taken.

Returns `true` if assessment is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_assessment_results``()`

Tests for the availability of an assessment rsults service.

Returns `true` if assessment results is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_item_lookup``()`

Tests if an item lookup service is supported.

Returns true if item lookup is supported, false otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_item_query``()`

Tests if an item query service is supported.

Returns `true` if item query is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_item_search``()`

Tests if an item search service is supported.

Returns `true` if item search is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_item_admin``()`

Tests if an item administrative service is supported.

Returns `true` if item admin is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_item_notification``()`

Tests if item notification is supported.

Messages may be sent when items are created, modified, or deleted.

Returns `true` if item notification is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_item_bank``()`

Tests if an item to bank lookup session is available.

Returns `true` if item bank lookup session is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_item_bank_assignment``()`

Tests if an item to bank assignment session is available.

Returns `true` if item bank assignment is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_assessment_lookup``()`

Tests if an assessment lookup service is supported.

An assessment lookup service defines methods to access assessments.

Returns true if assessment lookup is supported, false otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_assessment_query``()`

Tests if an assessment query service is supported.

Returns `true` if assessment query is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_assessment_admin()

Tests if an assessment administrative service is supported.

Returns `true` if assessment admin is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_assessment_bank()

Tests if an assessment to bank lookup session is available.

Returns `true` if assessment bank lookup session is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_assessment_bank_assignment()

Tests if an assessment to bank assignment session is available.

Returns `true` if assessment bank assignment is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_assessment_basic_authoring()

Tests if an assessment basic authoring session is available.

Returns `true` if assessment basic authoring is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_assessment_offered_lookup()

Tests if an assessment offered lookup service is supported.

Returns true if assessment offered lookup is supported, false otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_assessment_offered_query()

Tests if an assessment offered query service is supported.

Returns `true` if assessment offered query is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_assessment_offered_admin()

Tests if an assessment offered administrative service is supported.

Returns `true` if assessment offered admin is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_assessment_offered_bank()
Tests if an assessment offered to bank lookup session is available.

Returns `true` if assessment offered bank lookup session is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_assessment_offered_bank_assignment()
Tests if an assessment offered to bank assignment session is available.

Returns `true` if assessment offered bank assignment is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_assessment_taken_lookup()
Tests if an assessment taken lookup service is supported.

Returns `true` if assessment taken lookup is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_assessment_taken_query()
Tests if an assessment taken query service is supported.

Returns `true` if assessment taken query is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_assessment_taken_admin()
Tests if an assessment taken administrative service is supported which is used to instantiate an assessment offered.

Returns `true` if assessment taken admin is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_assessment_taken_bank()
Tests if an assessment taken to bank lookup session is available.

Returns `true` if assessment taken bank lookup session is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_assessment_taken_bank_assignment()
Tests if an assessment taken to bank assignment session is available.

Returns `true` if assessment taken bank assignment is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_bank_lookup()
Tests if a bank lookup service is supported.

A bank lookup service defines methods to access assessment banks.

Returns `true` if bank lookup is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_bank_query``()`
Tests if a bank query service is supported.

Returns `true` if bank query is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_bank_admin``()`
Tests if a banlk administrative service is supported.

Returns `true` if bank admin is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_bank_hierarchy``()`
Tests if a bank hierarchy traversal is supported.

Returns `true` if a bank hierarchy traversal is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_bank_hierarchy_design``()`
Tests if bank hierarchy design is supported.

Returns `true` if a bank hierarchy design is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`item_record_types`
Gets the supported `Item` record types.

Returns a list containing the supported `Item` record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

`item_search_record_types`
Gets the supported `Item` search record types.

Returns a list containing the supported `Item` search record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

`assessment_record_types`
Gets the supported `Assessment` record types.

Returns a list containing the supported `Assessment` record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

assessment_search_record_types

Gets the supported `Assessment` search record types.

Returns a list containing the supported assessment search record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

assessment_offered_record_types

Gets the supported `AssessmentOffered` record types.

Returns a list containing the supported `AssessmentOffered` record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

assessment_offered_search_record_types

Gets the supported `AssessmentOffered` search record types.

Returns a list containing the supported `AssessmentOffered` search record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

assessment_taken_record_types

Gets the supported `AssessmentTaken` record types.

Returns a list containing the supported `AssessmentTaken` record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

assessment_taken_search_record_types

Gets the supported `AssessmentTaken` search record types.

Returns a list containing the supported `AssessmentTaken` search record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

assessment_section_record_types

Gets the supported `AssessmentSection` record types.

Returns a list containing the supported `AssessmentSection` record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

bank_record_types

Gets the supported `Bank` record types.

Returns a list containing the supported `Bank` record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

bank_search_record_types

Gets the supported bank search record types.

Returns a list containing the supported `Bank` search record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

Assessment Manager

class `dlkit.services.assessment.`**`AssessmentManager`**(*proxy=None*)

Bases: `dlkit.osid.managers.OsidManager`, `dlkit.osid.sessions.OsidSession`, *`dlkit.services.assessment.AssessmentProfile`*

The assessment manager provides access to assessment sessions and provides interoperability tests for various aspects of this service.

The sessions included in this manager are:

- `MyAssessmentTakenSession`: a session to get taken or in progress assessments for the current agent
- `AssessmentSession`: a session to be assessed and examine assessments taken
- `AssessmentResultsSession`: a session to retrieve assessment results
- `ItemLookupSession`: a session to look up `Items`
- `ItemQuerySession` : a session to query `Items`
- `ItemSearchSession`: a session to search `Items`
- `ItemAdminSession`: a session to create, modify and delete `Items`
- `ItemNotificationSession`: a session to receive messages pertaining to `Item` changes
- `ItemBankSession`: a session for looking up item and bank mappings
- `ItemBankAssignmentSession`: a session for managing item and bank mappings
- `ItemSmartBankSession`: a session for managing dynamic banks
- `AssessmentLookupSession`: a session to look up `Assessments`
- `AssessmentQuerySession`: a session to query `Assessments`
- `AssessmentSearchSession`: a session to search `Assessments`
- `AssessmentAdminSession`: a session to create, modify and delete `Assessments`
- `AssessmentNotificationSession`: a session to receive messages pertaining to `Assessment` changes
- `AssessmentBankSession`: a session for looking up assessment and bank mappings
- `AssessmentBankAssignmentSession`: a session for managing assessment and bank mappings
- `AssessmentSmartBankSession`: a session for managing dynamic banks
- `AssessmentBasicAuthoringSession`: a session for making simple mappings of assessment items to assessments
- `AssessmentOfferedLookupSession`: a session to look up `AssessmentsOffered`
- `AssessmentOfferedQuerySession`: a session to query `AssessmentsOffered`
- `AssessmentOfferedSearchSession` : a session to search `AssessmentsOffered`
- `AssessmentOfferedAdminSession`: a session to create, modify and delete `AssessmentsOffered`
- `AssessmentOfferedNotificationSession`: a session to receive messages pertaining to `AssessmentOffered` changes

- `AssessmentOfferedBankSession`: a session for looking up assessments offered and bank mappings
- `AssessmentOfferedBankAssignmentSession`: a session for managing assessments offered and bank mappings
- `AssessmentOfferedSmartBankSession` : a session to manage dynamic banks of assessments offered
- `AssessmentTakenLookupSession`: a session to look up `Assessments`
- `AssessmentTakenQuerySession`: a session to query `Assessments`
- `AssessmentTakenSearchSession`: a session to search Assessments
- `AssessmentTakenAdminSession`: a session to create, modify and delete `AssessmentsTaken`
- `AssessmentTakenNotificationSession`: a session to receive messages pertaining to `AssessmentTaken` changes
- `AssessmentTakenBankSession`: a session for looking up assessments taken and bank mappings
- `AssessmenttTakenBankAssignmentSession`: a session for managing assessments taken and bank mappings
- `AssessmentTakenSmartBankSession`: a session to manage dynamic banks of assessments taken
- `BankLookupSession`: a session to lookup banks
- `BankQuerySession` : a session to query banks
- `BankSearchSession`: a session to search banks
- `BankAdminSession` : a session to create, modify and delete banks
- `BankNotificationSession` : a session to receive messages pertaining to `Bank` changes
- `BankHierarchySession` : a session to traverse the `Bank` hierarchy
- `BankHierarchyDesignSession` : a session to manage the `Bank` hierarchy

`assessment_authoring_manager`

Gets an `AssessmentAuthoringManager`.

Returns an `AssessmentAuthoringManager`

Return type `osid.assessment.authoring.AssessmentAuthoringManager`

Raise `OperationFailed` – unable to complete request

Raise `Unimplemented` – `supports_assessment_authoring() is false`

compliance: optional – This method must be implemented if ‘‘supports_assessment_authoring()‘‘ is true.

`assessment_batch_manager`

Gets an `AssessmentBatchManager`.

Returns an `AssessmentBatchManager`

Return type `osid.assessment.batch.AssessmentBatchManager`

Raise `OperationFailed` – unable to complete request

Raise `Unimplemented` – `supports_assessment_batch() is false`

compliance: optional – This method must be implemented if ‘‘supports_assessment_batch()‘‘ is true.

Bank Lookup Methods

`AssessmentManager.`**`can_lookup_banks`**`()`

Tests if this user can perform `Bank` lookups.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if lookup methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`use_comparative_bank_view`**`()`

The returns from the bank methods may omit or translate elements based on this session, such as assessment, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`AssessmentManager.`**`use_plenary_bank_view`**`()`

A complete view of the `Hierarchy` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`AssessmentManager.`**`get_bank`**`(`*bank_id*`)`

Gets the `Bank` specified by its `Id`.

In plenary mode, the exact `Id` is found or a `NotFound` results. Otherwise, the returned `Bank` may have a different `Id` than requested, such as the case where a duplicate `Id` was assigned to a `Bank` and retained for compatibility.

Parameters **`bank_id`** (`osid.id.Id`) – `Id` of the `Bank`

Returns the bank

Return type `osid.assessment.Bank`

Raise `NotFound` – `bank_id` not found

Raise `NullArgument` – `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method is must be implemented.

`AssessmentManager.`**`get_banks_by_ids`**`(`*bank_ids*`)`

Gets a `BankList` corresponding to the given `IdList`.

In plenary mode, the returned list contains all of the banks specified in the `Id` list, in the order of the list, including duplicates, or an error results if an `Id` in the supplied list is not found or inaccessible. Otherwise, inaccessible `Bank` objects may be omitted from the list and may present the elements in any order including returning a unique set.

Parameters **`bank_ids`** (`osid.id.IdList`) – the list of `Ids` to retrieve

Returns the returned `Bank` list

Return type `osid.assessment.BankList`

Raise `NotFound` – an `Id was` not found

Raise `NullArgument` – `bank_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`get_banks_by_genus_type`**(*bank_genus_type*)

Gets a `BankList` corresponding to the given bank genus `Type` which does not include banks of types derived from the specified `Type`.

In plenary mode, the returned list contains all known banks or an error results. Otherwise, the returned list may contain only those banks that are accessible through this session.

Parameters **`bank_genus_type`** (`osid.type.Type`) – a bank genus type

Returns the returned `Bank` list

Return type `osid.assessment.BankList`

Raise `NullArgument` – `bank_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`get_banks_by_parent_genus_type`**(*bank_genus_type*)

Gets a `BankList` corresponding to the given bank genus `Type` and include any additional banks with genus types derived from the specified `Type`.

In plenary mode, the returned list contains all known banks or an error results. Otherwise, the returned list may contain only those banks that are accessible through this session.

Parameters **`bank_genus_type`** (`osid.type.Type`) – a bank genus type

Returns the returned `Bank` list

Return type `osid.assessment.BankList`

Raise `NullArgument` – `bank_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`get_banks_by_record_type`**(*bank_record_type*)

Gets a `BankList` containing the given bank record `Type`.

In plenary mode, the returned list contains all known banks or an error results. Otherwise, the returned list may contain only those banks that are accessible through this session.

Parameters **`bank_record_type`** (`osid.type.Type`) – a bank record type

Returns the returned `Bank` list

Return type `osid.assessment.BankList`

Raise `NullArgument` – `bank_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`get_banks_by_provider`**`(`*resource_id*`)`

Gets a `BankList` from the given provider ````.

In plenary mode, the returned list contains all known banks or an error results. Otherwise, the returned list may contain only those banks that are accessible through this session.

Parameters **`resource_id`** (`osid.id.Id`) – a resource `Id`

Returns the returned `Bank` list

Return type `osid.assessment.BankList`

Raise `NullArgument` – `resource_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`banks`**

Gets all `Banks`.

In plenary mode, the returned list contains all known banks or an error results. Otherwise, the returned list may contain only those banks that are accessible through this session.

Returns a `BankList`

Return type `osid.assessment.BankList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

Bank Query Methods

`AssessmentManager.`**`can_search_banks`**`()`

Tests if this user can perform `Bank` searches.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer search operations to unauthorized users.

Returns `false` if search methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`bank_query`**

Gets a bank query.

Returns a bank query

Return type `osid.assessment.BankQuery`

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`get_banks_by_query`**`(`*bank_query*`)`

Gets a list of `Bank` objects matching the given bank query.

Parameters **`bank_query`** (`osid.assessment.BankQuery`) – the bank query

Returns the returned `BankList`

Return type `osid.assessment.BankList`

Raise `NullArgument` – `bank_query` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – `bank_query` is not of this service

compliance: mandatory – This method must be implemented.

Bank Admin Methods

`AssessmentManager.`**`can_create_banks`**`()`

Tests if this user can create `Banks`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known creating a `Bank` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer create operations to unauthorized users.

Returns `false` if `Bank` creation is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`can_create_bank_with_record_types`**`(`*bank_record_types*`)`

Tests if this user can create a single `Bank` using the desired record types.

While `AssessmentManager.getBankRecordTypes()` can be used to examine which records are supported, this method tests which record(s) are required for creating a specific `Bank`. Providing an empty array tests if a `Bank` can be created with no records.

Parameters **`bank_record_types`** (`osid.type.Type[]`) – array of bank record types

Returns `true` if `Bank` creation using the specified `Types` is supported, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `bank_record_types` is `null`

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`get_bank_form_for_create`**`(`*bank_record_types*`)`

Gets the bank form for creating new banks.

A new form should be requested for each create transaction.

Parameters **`bank_record_types`** (`osid.type.Type[]`) – array of bank record types to be included in the create operation or an empty list if none

Returns the bank form

Return type `osid.assessment.BankForm`

Raise `NullArgument` – `bank_record_types` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – unable to get form for requested record types

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`create_bank`**`(`*bank_form*`)`

Creates a new `Bank`.

Parameters **`bank_form`** (`osid.assessment.BankForm`) – the form for this `Bank`

Returns the new `Bank`

Return type `osid.assessment.Bank`

Raise `IllegalState` – `bank_form` already used in a create transaction

Raise `InvalidArgument` – one or more of the form elements is invalid

Raise `NullArgument` – `bank_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – `bank_form` did not originate from `get_bank_form_for_create()`

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`can_update_banks`**`()`

Tests if this user can update `Banks`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating a `Bank` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer update operations to unauthorized users.

Returns `false` if `Bank` modification is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`get_bank_form_for_update`**`(`*bank_id*`)`

Gets the bank form for updating an existing bank.

A new bank form should be requested for each update transaction.

Parameters **`bank_id`** (`osid.id.Id`) – the `Id` of the `Bank`

Returns the bank form

Return type `osid.assessment.BankForm`

Raise `NotFound` – `bank_id` is not found

Raise `NullArgument` – `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`update_bank`**`(`*bank_form*`)`

Updates an existing bank.

Parameters **`bank_form`** (`osid.assessment.BankForm`) – the form containing the elements to be updated

Raise `IllegalState` – `bank_form` already used in an update transaction

Raise `InvalidArgument` – the form contains an invalid value

Raise `NullArgument` – `bank_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – `bank_form` did not originate from `get_bank_form_for_update()`

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`can_delete_banks`**`()`

Tests if this user can delete banks.

A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting a `Bank` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer delete operations to unauthorized users.

Returns `false` if `Bank` deletion is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`delete_bank`**`(`*bank_id*`)`

Deletes a `Bank`.

Parameters **`bank_id`** (`osid.id.Id`) – the `Id` of the `Bank` to remove

Raise `NotFound` – `bank_id` not found

Raise `NullArgument` – `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`can_manage_bank_aliases`**`()`

Tests if this user can manage `Id` aliases for `Banks`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known changing an alias will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer alias operations to an unauthorized user.

Returns `false` if `Bank` aliasing is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`alias_bank`**`(`*bank_id*, *alias_id*`)`

Adds an `Id` to a `Bank` for the purpose of creating compatibility.

The primary `Id` of the `Bank` is determined by the provider. The new `Id` is an alias to the primary `Id`. If the alias is a pointer to another bank, it is reassigned to the given bank `Id`.

Parameters

- **`bank_id`** (`osid.id.Id`) – the `Id` of a `Bank`
- **`alias_id`** (`osid.id.Id`) – the alias `Id`

Raise `AlreadyExists` – `alias_id` is in use as a primary `Id`

Raise `NotFound` – `bank_id` not found

Raise `NullArgument` – `bank_id` or `alias_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

Bank Hierarchy Methods

`AssessmentManager.`**`bank_hierarchy_id`**

Gets the hierarchy `Id` associated with this session.

Returns the hierarchy `Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`bank_hierarchy`**

Gets the hierarchy associated with this session.

Returns the hierarchy associated with this session

Return type `osid.hierarchy.Hierarchy`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – assessment failure

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`can_access_bank_hierarchy`**`()`

Tests if this user can perform hierarchy queries.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations.

Returns `false` if hierarchy traversal methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`use_comparative_bank_view`**`()`

The returns from the bank methods may omit or translate elements based on this session, such as assessment, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`AssessmentManager.`**`use_plenary_bank_view`**`()`

A complete view of the `Hierarchy` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`AssessmentManager.`**`root_bank_ids`**

Gets the root bank `Ids` in this hierarchy.

Returns the root bank `Ids`

Return type `osid.id.IdList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`root_banks`**

Gets the root banks in this bank hierarchy.

Returns the root banks

Return type `osid.assessment.BankList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method is must be implemented.

`AssessmentManager.`**`has_parent_banks`**(*bank_id*)

Tests if the `Bank` has any parents.

Parameters **`bank_id`** (`osid.id.Id`) – a bank `Id`

Returns `true` if the bank has parents, `false` otherwise

Return type `boolean`

Raise `NotFound` – `bank_id` is not found

Raise `NullArgument` – `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`is_parent_of_bank`**(*id_*, *bank_id*)

Tests if an `Id` is a direct parent of a bank.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`bank_id`** (`osid.id.Id`) – the `Id` of a bank

Returns `true` if this `id` is a parent of `bank_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `bank_id` is not found

Raise `NullArgument` – `id` or `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented. implementation notes: If `id` not found return `false`.

`AssessmentManager.`**`get_parent_bank_ids`**`(`*bank_id*`)`

Gets the parent `Ids` of the given bank.

Parameters **`bank_id`** (`osid.id.Id`) – a bank `Id`

Returns the parent `Ids` of the bank

Return type `osid.id.IdList`

Raise `NotFound` – `bank_id` is not found

Raise `NullArgument` – `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`get_parent_banks`**`(`*bank_id*`)`

Gets the parents of the given bank.

Parameters **`bank_id`** (`osid.id.Id`) – a bank `Id`

Returns the parents of the bank

Return type `osid.assessment.BankList`

Raise `NotFound` – `bank_id` is not found

Raise `NullArgument` – `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`is_ancestor_of_bank`**`(`*id_*, *bank_id*`)`

Tests if an `Id` is an ancestor of a bank.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`bank_id`** (`osid.id.Id`) – the `Id` of a bank

Returns `true` if this `id` is an ancestor of `bank_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `bank_id` is not found

Raise `NullArgument` – `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` not found return `false`.

`AssessmentManager.`**`has_child_banks`**`(`*bank_id*`)`

Tests if a bank has any children.

Parameters **`bank_id`** (`osid.id.Id`) – a `bank_id`

Returns `true` if the `bank_id` has children, `false` otherwise

Return type `boolean`

Raise `NotFound` – `bank_id` is not found

Raise `NullArgument` – `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`is_child_of_bank`**`(`*id_*, *bank_id*`)`

Tests if a bank is a direct child of another.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`bank_id`** (`osid.id.Id`) – the `Id` of a bank

Returns `true` if the `id` is a child of `bank_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `bank_id` not found

Raise `NullArgument` – `bank_id` or `id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` not found return `false`.

`AssessmentManager.`**`get_child_bank_ids`**`(`*bank_id*`)`

Gets the child `Ids` of the given bank.

Parameters **`bank_id`** (`osid.id.Id`) – the `Id` to query

Returns the children of the bank

Return type `osid.id.IdList`

Raise `NotFound` – `bank_id` is not found

Raise `NullArgument` – `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`get_child_banks`**`(`*bank_id*`)`

Gets the children of the given bank.

Parameters **`bank_id`** (`osid.id.Id`) – the `Id` to query

Returns the children of the bank

Return type `osid.assessment.BankList`

Raise `NotFound` – `bank_id` is not found

Raise `NullArgument` – `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`is_descendant_of_bank`**(*id_*, *bank_id*)

Tests if an `Id` is a descendant of a bank.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`bank_id`** (`osid.id.Id`) – the `Id` of a bank

Returns `true` if the `id` is a descendant of the `bank_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `bank_id` not found

Raise `NullArgument` – `bank_id` or `id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` is not found return `false`.

`AssessmentManager.`**`get_bank_node_ids`**(*bank_id*, *ancestor_levels*, *descendant_levels*, *include_siblings*)

Gets a portion of the hierarchy for the given bank.

Parameters

- **`bank_id`** (`osid.id.Id`) – the `Id` to query
- **`ancestor_levels`** (`cardinal`) – the maximum number of ancestor levels to include. A value of 0 returns no parents in the node.
- **`descendant_levels`** (`cardinal`) – the maximum number of descendant levels to include. A value of 0 returns no children in the node.
- **`include_siblings`** (`boolean`) – `true` to include the siblings of the given node, `false` to omit the siblings

Returns a bank node

Return type `osid.hierarchy.Node`

Raise `NotFound` – `bank_id` is not found

Raise `NullArgument` – `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`get_bank_nodes`**(*bank_id*, *ancestor_levels*, *descendant_levels*, *include_siblings*)

Gets a portion of the hierarchy for the given bank.

Parameters

- **`bank_id`** (`osid.id.Id`) – the `Id` to query
- **`ancestor_levels`** (`cardinal`) – the maximum number of ancestor levels to include. A value of 0 returns no parents in the node.

- **descendant_levels** (`cardinal`) – the maximum number of descendant levels to include. A value of 0 returns no children in the node.
- **include_siblings** (`boolean`) – `true` to include the siblings of the given node, `false` to omit the siblings

Returns a bank node

Return type `osid.assessment.BankNode`

Raise `NotFound` – `bank_id` is not found

Raise `NullArgument` – `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Bank Hierarchy Design Methods

`AssessmentManager.`**`bank_hierarchy_id`**

Gets the hierarchy `Id` associated with this session.

Returns the hierarchy `Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`bank_hierarchy`**

Gets the hierarchy associated with this session.

Returns the hierarchy associated with this session

Return type `osid.hierarchy.Hierarchy`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – assessment failure

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`can_modify_bank_hierarchy`**`()`

Tests if this user can change the hierarchy.

A return of true does not guarantee successful authorization. A return of false indicates that it is known performing any update will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer these operations to an unauthorized user.

Returns `false` if changing this hierarchy is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`add_root_bank`**`(`*bank_id*`)`

Adds a root bank.

Parameters **bank_id** (`osid.id.Id`) – the `Id` of a bank

Raise `AlreadyExists` – `bank_id` is already in hierarchy

Raise `NotFound` – `bank_id` not found

Raise `NullArgument` – `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`remove_root_bank`**(*bank_id*)

Removes a root bank from this hierarchy.

Parameters **`bank_id`** (`osid.id.Id`) – the `Id` of a bank

Raise `NotFound` – `bank_id` not a parent of `child_id`

Raise `NullArgument` – `bank_id` or `child_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`add_child_bank`**(*bank_id*, *child_id*)

Adds a child to a bank.

Parameters

- **`bank_id`** (`osid.id.Id`) – the `Id` of a bank
- **`child_id`** (`osid.id.Id`) – the `Id` of the new child

Raise `AlreadyExists` – `bank_id` is already a parent of `child_id`

Raise `NotFound` – `bank_id` or `child_id` not found

Raise `NullArgument` – `bank_id` or `child_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`remove_child_bank`**(*bank_id*, *child_id*)

Removes a child from a bank.

Parameters

- **`bank_id`** (`osid.id.Id`) – the `Id` of a bank
- **`child_id`** (`osid.id.Id`) – the `Id` of the new child

Raise `NotFound` – `bank_id` not parent of `child_id`

Raise `NullArgument` – `bank_id` or `child_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`AssessmentManager.`**`remove_child_banks`**(*bank_id*)

Removes all children from a bank.

Parameters **`bank_id`** (`osid.id.Id`) – the `Id` of a bank

Raise `NotFound` – `bank_id` is not in hierarchy

Raise `NullArgument` – `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

Bank

Bank

class `dlkit.services.assessment.`**`Bank`**(*provider_manager*, *catalog*, *runtime*, *proxy*, ***kwargs*)

Bases: *`dlkit.osid.objects.OsidCatalog`*, `dlkit.osid.sessions.OsidSession`

A bank defines a collection of assessments and items.

`get_bank_record`(*bank_record_type*)

Gets the bank record corresponding to the given `Bank` record `Type`.

This method is used to retrieve an object implementing the requested record. The `bank_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(bank_record_type)` is `true` .

Parameters **`bank_record_type`** (`osid.type.Type`) – a bank record type

Returns the bank record

Return type `osid.assessment.records.BankRecord`

Raise `NullArgument` – `bank_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(bank_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Assessment Methods

`Bank.`**`bank_id`**

Gets the `Bank Id` associated with this session.

Returns the `Bank Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Bank.`**`bank`**

Gets the `Bank` associated with this session.

Returns the `Bank` associated with this session

Return type `osid.assessment.Bank`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`can_take_assessments`**`()`

Tests if this user can take this assessment section.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer assessment operations to unauthorized users.

Returns `false` if assessment methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`has_assessment_begun`**`(`*assessment_taken_id*`)`

Tests if this assessment has started.

An assessment begins from the designated start time if a start time is defined. If no start time is defined the assessment may begin at any time. Assessment sections cannot be accessed if the return for this method is `false`.

Parameters **`assessment_taken_id`** (`osid.id.Id`) – `Id` of the `AssessmentTaken`

Returns `true` if this assessment has begun, `false` otherwise

Return type `boolean`

Raise `NotFound` – `assessment_taken_id` is not found

Raise `NullArgument` – `assessment_taken_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`is_assessment_over`**`(`*assessment_taken_id*`)`

Tests if this assessment is over.

An assessment is over if `finished_assessment()` is invoked or the designated finish time has expired.

Parameters **`assessment_taken_id`** (`osid.id.Id`) – `Id` of the `AssessmentTaken`

Returns `true` if this assessment is over, `false` otherwise

Return type `boolean`

Raise `NotFound` – `assessment_taken_id` is not found

Raise `NullArgument` – `assessment_taken_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`requires_synchronous_sections`**`(`*assessment_taken_id*`)`

Tests if synchronous sections are required.

This method should be checked to determine if all sections are available when requested, or the next sections becomes available only after the previous section is complete.

There are two methods for retrieving sections. One is using the built-in hasNextSection() and getNextSection() methods. In synchronous mode, hasNextSection() is false until the current section is completed. In asynchronous mode, `has_next_section()` returns true until the end of the assessment.

`AssessmentSections` may also be accessed via an `AssessmentSectionList`. If syncronous sections are required, `AssessmentSectionList.available() == 0` and `AssessmentSectionList.getNextQuestion()` blocks until the section is complete. `AssessmentSectionList.hasNext()` is always true until the end of the assessment is reached.

Parameters **`assessment_taken_id`** (`osid.id.Id`) – `Id` of the `AssessmentTaken`

Returns `true` if this synchronous sections are required, `false` otherwise

Return type `boolean`

Raise `NotFound` – `assessment_taken_id` is not found

Raise `NullArgument` – `assessment_taken_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_first_assessment_section`**(*assessment_taken_id*)

Gets the first assessment section in this assesment.

All assessments have at least one `AssessmentSection`.

Parameters **`assessment_taken_id`** (`osid.id.Id`) – `Id` of the `AssessmentTaken`

Returns the first assessment section

Return type `osid.assessment.AssessmentSection`

Raise `IllegalState` – `has_assessment_begun()` is `false`

Raise `NotFound` – `assessment_taken_id` is not found

Raise `NullArgument` – `assessment_taken_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`has_next_assessment_section`**(*assessment_section_id*)

Tests if there is a next assessment section in the assessment following the given assessment section `Id`.

Parameters **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`

Returns `true` if there is a next section, `false` otherwise

Return type `boolean`

Raise `IllegalState` – `has_assessment_begun()` is `false`

Raise `NotFound` – `assessment_taken_id` is not found

Raise `NullArgument` – `assessment_taken_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_next_assessment_section`**(*assessment_section_id*)

Gets the next assessemnt section following the given assesment section.

Parameters **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`

Returns the next section

Return type `osid.assessment.AssessmentSection`

Raise `IllegalState` – `has_next_assessment_section()` is `false`

Raise `NotFound` – `assessment_section_id` is not found

Raise `NullArgument` – `assessment_section_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`has_previous_assessment_section`**(*assessment_section_id*)

Tests if there is a previous assessment section in the assessment following the given assessment section `Id`.

Parameters **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`

Returns `true` if there is a previous assessment section, `false` otherwise

Return type `boolean`

Raise `IllegalState` – `has_assessment_begun()` is `false`

Raise `NotFound` – `assessment_section_id` is not found

Raise `NullArgument` – `assessment_section_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_previous_assessment_section`**(*assessment_section_id*)

Gets the next assessemnt section following the given assesment section.

Parameters **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`

Returns the previous assessment section

Return type `osid.assessment.AssessmentSection`

Raise `IllegalState` – `has_next_assessment_section()` is `false`

Raise `NotFound` – `assessment_section_id` is not found

Raise `NullArgument` – `assessment_section_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessment_section`**`(`*assessment_section_id*`)`

Gets an assessemnts section by `Id`.

Parameters **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`

Returns the assessment section

Return type `osid.assessment.AssessmentSection`

Raise `IllegalState` – `has_assessment_begun()` is `false`

Raise `NotFound` – `assessment_section_id` is not found

Raise `NullArgument` – `assessment_section_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessment_sections`**`(`*assessment_taken_id*`)`

Gets the assessment sections of this assessment.

Parameters **`assessment_taken_id`** (`osid.id.Id`) – `Id` of the `AssessmentTaken`

Returns the list of assessment sections

Return type `osid.assessment.AssessmentSectionList`

Raise `IllegalState` – `has_assessment_begun()` is `false`

Raise `NotFound` – `assessment_taken_id` is not found

Raise `NullArgument` – `assessment_taken_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`is_assessment_section_complete`**`(`*assessment_section_id*`)`

Tests if the all responses have been submitted to this assessment section.

If `is_assessment_section_complete()` is false, then `get_unanswered_questions()` may return a list of questions that can be submitted.

Parameters **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`

Returns `true` if this assessment section is complete, `false` otherwise

Return type `boolean`

Raise `IllegalState` – `is_assessment_over()` is `true`

Raise `NotFound` – `assessment_section_id` is not found

Raise `NullArgument` – `assessment_section_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bank.`**`get_incomplete_assessment_sections`**(*assessment_taken_id*)

Gets the incomplete assessment sections of this assessment.

Parameters **`assessment_taken_id`** (`osid.id.Id`) – `Id` of the `AssessmentTaken`

Returns the list of incomplete assessment sections

Return type `osid.assessment.AssessmentSectionList`

Raise `IllegalState` – `has_assessment_begun()` is `false`

Raise `NotFound` – `assessment_taken_id` is not found

Raise `NullArgument` – `assessment_taken_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`has_assessment_section_begun`**(*assessment_section_id*)

Tests if this assessment section has started.

A section begins from the designated start time if a start time is defined. If no start time is defined the section may begin at any time. Assessment items cannot be accessed or submitted if the return for this method is `false`.

Parameters **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`

Returns `true` if this assessment section has begun, `false` otherwise

Return type `boolean`

Raise `IllegalState` – `has_assessment_begun()` is `false or is_assessment_over()` is `true`

Raise `NotFound` – `assessment_section_id` is not found

Raise `NullArgument` – `assessment_section_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bank.`**`is_assessment_section_over`**(*assessment_section_id*)

Tests if this assessment section is over.

An assessment section is over if new or updated responses can not be submitted such as the designated finish time has expired.

Parameters **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`

Returns `true` if this assessment is over, `false` otherwise

Return type `boolean`

Raise `IllegalState` – `has_assessmen_sectiont_begun()` is `false` or `is_assessment_section_over()` is `true`

Raise `NotFound` – `assessment_section_id` is not found

Raise `NullArgument` – `assessment_section_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bank.`**`requires_synchronous_responses`**(*assessment_section_id*)

Tests if synchronous responses are required in this assessment section.

This method should be checked to determine if all items are available when requested, or the next item becomes available only after the response to the current item is submitted.

There are two methods for retrieving questions. One is using the built-in `has_next_question()` and `get_next_question()` methods. In synchronous mode, `has_next_question()` is `false` until the response for the current question is submitted. In asynchronous mode, `has_next_question()` returns `true` until the end of the assessment.

`Questions` may also be accessed via a `QuestionList`. If syncronous responses are required, `QuestionList.available() == 0` and `QuestionList.getNextQuestion()` blocks until the response is submitted. `QuestionList.hasNext()` is always true until the end of the assessment is reached.

Parameters **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`

Returns `true` if this synchronous responses are required, `false` otherwise

Return type `boolean`

Raise `IllegalState` – `has_assessment_begun()` is `false` or `is_assessment_over()` is `true`

Raise `NotFound` – `assessment_section_id` is not found

Raise `NullArgument` – `assessment_section_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bank.`**`get_first_question`**(*assessment_section_id*)

Gets the first question in this assesment section.

Parameters **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`

Returns the first question

Return type `osid.assessment.Question`

Raise `IllegalState – has_assessment_section_begun() is false or is_assessment_section_over() is true`

Raise `NotFound` – `assessment_section_id` is not found

Raise `NullArgument` – `assessment_section_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`has_next_question`**(*assessment_section_id*, *item_id*)

Tests if there is a next question following the given question `Id`.

Parameters

- **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`
- **`item_id`** (`osid.id.Id`) – `Id` of the `Item`

Returns `true` if there is a next question, `false` otherwise

Return type `boolean`

Raise `IllegalState` – `has_assessment_section_begun()` is `false` or `is_assessment_section_over()` is `true`

Raise `NotFound` – `assessment_section_id` or `item_id` is not found, or `item_id` not part of `assessment_section_id`

Raise `NullArgument` – `assessment_section_id` or `item_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_next_question`**(*assessment_section_id*, *item_id*)

Gets the next question in this assesment section.

Parameters

- **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`
- **`item_id`** (`osid.id.Id`) – `Id` of the `Item`

Returns the next question

Return type `osid.assessment.Question`

Raise `IllegalState` – `has_next_question()` is `false`

Raise `NotFound` – `assessment_section_id` or `item_id` is not found, or `item_id` not part of `assessment_section_id`

Raise `NullArgument` – `assessment_section_id` or `item_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`has_previous_question`**(*assessment_section_id*, *item_id*)

Tests if there is a previous question preceeding the given question `Id`.

Parameters

- **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`
- **`item_id`** (`osid.id.Id`) – `Id` of the `Item`

Returns `true` if there is a previous question, `false` otherwise

Return type `boolean`

Raise `IllegalState` – `has_assessment_section_begun()` is `false` or `is_assessment_section_over()` is `true`

Raise `NotFound` – `assessment_section_id` or `item_id` is not found, or `item_id` not part of `assessment_section_id`

Raise `NullArgument` – `assessment_section_id` or `item_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_previous_question`**(*assessment_section_id*, *item_id*)

Gets the previous question in this assesment section.

Parameters

- **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`
- **`item_id`** (`osid.id.Id`) – `Id` of the `Item`

Returns the previous question

Return type `osid.assessment.Question`

Raise `IllegalState` – `has_previous_question()` is `false`

Raise `NotFound` – `assessment_section_id` or `item_id` is not found, or `item_id` not part of `assessment_section_id`

Raise `NullArgument` – `assessment_section_id` or `item_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_question`**(*assessment_section_id*, *item_id*)

Gets the `Question` specified by its `Id`.

Parameters

- **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`
- **`item_id`** (`osid.id.Id`) – `Id` of the `Item`

Returns the returned `Question`

Return type `osid.assessment.Question`

Raise `IllegalState` – `has_assessment_section_begun()` is `false` or `is_assessment_section_over()` is `true`

Raise `NotFound` – `assessment_section_id` or `item_id` is not found, or `item_id` not part of `assessment_section_id`

Raise `NullArgument` – `assessment_section_id` or `item_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_questions`**(*assessment_section_id*)

Gets the questions of this assessment section.

Parameters **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`

Returns the list of assessment questions

Return type `osid.assessment.QuestionList`

Raise `IllegalState` – `has_assessment_section_begun() is false or is_assessment_section_over() is true`

Raise `NotFound` – `assessment_section_id` is not found

Raise `NullArgument` – `assessment_section_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_response_form`**(*assessment_section_id*, *item_id*)

Gets the response form for submitting an answer.

Parameters

- **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`
- **`item_id`** (`osid.id.Id`) – `Id` of the `Item`

Returns an answer form

Return type `osid.assessment.AnswerForm`

Raise `IllegalState` – `has_assessment_section_begun()` is `false or is_assessment_section_over()` is `true`

Raise `NotFound` – `assessment_section_id` or `item_id` is not found, or `item_id` not part of `assessment_section_id`

Raise `NullArgument` – `assessment_section_id` or `item_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`submit_response`**(*assessment_section_id*, *item_id*, *answer_form*)

Submits an answer to an item.

Parameters

- **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`
- **`item_id`** (`osid.id.Id`) – `Id` of the `Item`
- **`answer_form`** (`osid.assessment.AnswerForm`) – the response

Raise `IllegalState` – `has_assessment_section_begun()` is `false` or `is_assessment_section_over()` is `true`

Raise `InvalidArgument` – one or more of the elements in the form is invalid

Raise `NotFound` – `assessment_section_id` or `item_id` is not found, or `item_id` not part of `assessment_section_id`

Raise `NullArgument` – `assessment_section_id`, `item_id`, or `answer_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `answer_form` is not of this service

compliance: mandatory – This method must be implemented.

`Bank.`**`skip_item`**(*assessment_section_id*, *item_id*)

Skips an item.

Parameters

- **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`
- **`item_id`** (`osid.id.Id`) – `Id` of the `Item`

Raise `IllegalState` – `has_assessment_section_begun()` is `false` or `is_assessment_section_over()` is `true`

Raise `NotFound` – `assessment_section_id` or `item_id` is not found, or `item_id` not part of `assessment_section_id`

Raise `NullArgument` – `assessment_section_id` or `item_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bank.`**`is_question_answered`**(*assessment_section_id*, *item_id*)

Tests if the given item has a response.

Parameters

- **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`
- **`item_id`** (`osid.id.Id`) – `Id` of the `Item`

Returns `true` if this item has a response, `false` otherwise

Return type `boolean`

Raise `IllegalState` – `has_assessment_section_begun()` is `false` or `is_assessment_section_over()` is `true`

Raise `NotFound` – `assessment_section_id or item_id is not found, or item_id not part of assessment_section_id`

Raise `NullArgument` – `assessment_section_id or item_id is null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bank.`**`get_unanswered_questions`**(*assessment_section_id*)

Gets the unanswered questions of this assessment section.

Parameters **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`

Returns the list of questions with no rsponses

Return type `osid.assessment.QuestionList`

Raise `IllegalState` – `has_assessment_section_begun()` is `false` or `is_assessment_section_over()` is `true`

Raise `NotFound` – `assessment_section_id` is not found

Raise `NullArgument` – `assessment_section_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`has_unanswered_questions`**(*assessment_section_id*)

Tests if there are unanswered questions in this assessment section.

Parameters **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`

Returns `true` if there are unanswered questions, `false` otherwise

Return type `boolean`

Raise `IllegalState` – `has_assessment_section_begun()` is `false` or `is_assessment_section_over()` is `true`

Raise `NotFound` – `assessment_section_id` is not found

Raise `NullArgument` – `assessment_section_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_first_unanswered_question`**(*assessment_section_id*)

Gets the first unanswered question in this assesment section.

Parameters **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`

Returns the first unanswered question

Return type `osid.assessment.Question`

Raise `IllegalState` – `has_unanswered_questions()` is `false`

Raise `NotFound` – `assessment_section_id` is not found

Raise `NullArgument` – `assessment_section_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`has_next_unanswered_question`**(*assessment_section_id*, *item_id*)

Tests if there is a next unanswered question following the given question `Id`.

Parameters

- **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`
- **`item_id`** (`osid.id.Id`) – `Id` of the `Item`

Returns `true` if there is a next unanswered question, `false` otherwise

Return type `boolean`

Raise `IllegalState` – `has_assessment_section_begun() is false or is_assessment_section_over() is true`

Raise `NotFound` – `assessment_section_id or item_id is not found, or item_id not part of assessment_section_id`

Raise `NullArgument` – `assessment_section_id or item_id is null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_next_unanswered_question`**(*assessment_section_id*, *item_id*)

Gets the next unanswered question in this assesment section.

Parameters

- **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`
- **`item_id`** (`osid.id.Id`) – `Id` of the `Item`

Returns the next unanswered question

Return type `osid.assessment.Question`

Raise `IllegalState` – `has_next_unanswered_question()` is `false`

Raise `NotFound` – `assessment_section_id or item_id is not found, or item_id not part of assessment_section_id`

Raise `NullArgument` – `assessment_section_id or item_id is null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`has_previous_unanswered_question`**(*assessment_section_id*, *item_id*)

Tests if there is a previous unanswered question preceeding the given question `Id`.

Parameters

- **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`
- **`item_id`** (`osid.id.Id`) – `Id` of the `Item`

Returns `true` if there is a previous unanswered question, `false` otherwise

Return type `boolean`

Raise `IllegalState` – `has_assessment_section_begun()` is `false` or `is_assessment_section_over()` is `true`

Raise `NotFound` – `assessment_section_id` or `item_id` is not found, or `item_id` not part of `assessment_section_id`

Raise `NullArgument` – `assessment_section_id` or `item_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_previous_unanswered_question`**(*assessment_section_id*, *item_id*)

Gets the previous unanswered question in this assesment section.

Parameters

- **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`
- **`item_id`** (`osid.id.Id`) – `Id` of the `Item`

Returns the previous unanswered question

Return type `osid.assessment.Question`

Raise `IllegalState` – `has_previous_unanswered_question()` is `false`

Raise `NotFound` – `assessment_section_id` or `item_id` is not found, or `item_id` not part of `assessment_section_id`

Raise `NullArgument` – `assessment_section_id` or `item_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_response`**(*assessment_section_id*, *item_id*)

Gets the submitted response to the associated item.

Parameters

- **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`
- **`item_id`** (`osid.id.Id`) – `Id` of the `Item`

Returns the response

Return type `osid.assessment.Response`

Raise `IllegalState` – `has_assessment_section_begun()` is `false` or `is_assessment_section_over()` is `true`

Raise `NotFound` – `assessment_section_id` or `item_id` is not found, or `item_id` not part of `assessment_section_id`

Raise `NullArgument` – `assessment_section_id` or `item_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bank.`**`get_responses`**(*assessment_taken_id*)

Gets the submitted responses.

Parameters **`assessment_taken_id`** (`osid.id.Id`) – `Id` of the `AssessmentTaken`

Returns the submitted answers

Return type `osid.assessment.ResponseList`

Raise `NotFound` – `assessment_taken_id` is not found

Raise `NullArgument` – `assessment_taken_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bank.`**`clear_response`**(*assessment_section_id*, *item_id*)

Clears the response to an item The item appears as unanswered.

If no response exists, the method simply returns.

Parameters

- **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`
- **`item_id`** (`osid.id.Id`) – `Id` of the `Item`

Raise `IllegalState` – `has_assessment_section_begun() is false or is_assessment_section_over() is true`

Raise `NotFound` – `assessment_section_id or item_id is not found, or item_id not part of assessment_section_id`

Raise `NullArgument` – `assessment_section_id or item_id is null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bank.`**`finish_assessment_section`**(*assessment_section_id*)

Indicates an assessment section is complete.

Finished sections may or may not allow new or updated responses.

Parameters **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`

Raise `IllegalState` – `has_assessment_section_begun()` is `false` or `is_assessment_section_over()` is `true`

Raise `NotFound` – `assessment_section_id` is not found

Raise `NullArgument` – `assessment_section_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bank.`**`is_answer_available`**(*assessment_section_id*, *item_id*)
Tests if an answer is available for the given item.

Parameters

- **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`
- **`item_id`** (`osid.id.Id`) – `Id` of the `Item`

Returns `true` if an answer are available, `false` otherwise

Return type `boolean`

Raise `NotFound` – `assessment_section_id or item_id is not found, or item_id not part of assessment_section_id`

Raise `NullArgument` – `assessment_section_id or item_id is null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bank.`**`get_answers`**(*assessment_section_id*, *item_id*)
Gets the acceptable answers to the associated item.

Parameters

- **`assessment_section_id`** (`osid.id.Id`) – `Id` of the `AssessmentSection`
- **`item_id`** (`osid.id.Id`) – `Id` of the `Item`

Returns the answers

Return type `osid.assessment.AnswerList`

Raise `IllegalState` – `is_answer_available()` is `false`

Raise `NotFound` – `assessment_section_id or item_id is not found, or item_id not part of assessment_section_id`

Raise `NullArgument` – `assessment_section_id or item_id is null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bank.`**`finish_assessment`**(*assessment_taken_id*)
Indicates the entire assessment is complete.

Parameters **`assessment_taken_id`** (`osid.id.Id`) – `Id` of the `AssessmentTaken`

Raise `IllegalState` – `has_begun()` is `false or is_over()` is `true`

Raise `NotFound` – `assessment_taken_id` is not found

Raise `NullArgument` – `assessment_taken_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Assessment Results Methods

`Bank.`**`bank_id`**

Gets the `Bank Id` associated with this session.

Returns the `Bank Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Bank.`**`bank`**

Gets the `Bank` associated with this session.

Returns the `Bank` associated with this session

Return type `osid.assessment.Bank`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`can_access_assessment_results`**`()`

Tests if this user can take this assessment.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer assessment operations to unauthorized users.

Returns `false` if assessment methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`get_items`**`()`

Gets all `Items`.

In plenary mode, the returned list contains all known items or an error results. Otherwise, the returned list may contain only those items that are accessible through this session.

Returns a list of `Items`

Return type `osid.assessment.ItemList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_responses`**`(`*assessment_taken_id*`)`

Gets the submitted responses.

Parameters **`assessment_taken_id`** (`osid.id.Id`) – `Id` of the `AssessmentTaken`

Returns the submitted answers

Return type `osid.assessment.ResponseList`

Raise `NotFound` – `assessment_taken_id` is not found

Raise `NullArgument` – `assessment_taken_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bank.`**`are_results_available`**`(`*assessment_taken_id*`)`

Tests if the results are available for this assessment.

Parameters **`assessment_taken_id`** (`osid.id.Id`) – `Id` of the `AssessmentTaken`

Returns `true` if results are available, `false` otherwise

Return type `boolean`

Raise `NotFound` – `assessment_taken_id` is not found

Raise `NullArgument` – `assessment_taken_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bank.`**`get_grade_entries`**`(`*assessment_taken_id*`)`

Gets a list of grade entries for this assessment.

Each grade entry may indicate a grade or score input by multiple graders.

Parameters **`assessment_taken_id`** (`osid.id.Id`) – `Id` of the `AssessmentTaken`

Returns a list of grade entries

Return type `osid.grading.GradeEntryList`

Raise `IllegalState` – `are_results_available()` is `false`

Raise `NotFound` – `assessment_taken_id` is not found

Raise `NullArgument` – `assessment_taken_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Item Lookup Methods

`Bank.`**`bank_id`**

Gets the `Bank Id` associated with this session.

Returns the `Bank Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Bank.`**`bank`**

Gets the `Bank` associated with this session.

Returns the `Bank` associated with this session

Return type `osid.assessment.Bank`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`can_lookup_items`**`()`

Tests if this user can perform `Item` lookups.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations.

Returns `false` if lookup methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`use_comparative_item_view`**`()`

The returns from the lookup methods may omit or translate elements based on this session, such as assessment, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`Bank.`**`use_plenary_item_view`**`()`

A complete view of the `Item` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`Bank.`**`use_federated_bank_view`**`()`

Federates the view for methods in this session.

A federated view will include assessments taken in banks which are children of this bank in the bank hierarchy.

compliance: mandatory – This method is must be implemented.

`Bank.`**`use_isolated_bank_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts searches to this bank only.

compliance: mandatory – This method is must be implemented.

`Bank.`**`get_item`**`(`*item_id*`)`

Gets the `Item` specified by its `Id`.

In plenary mode, the exact `Id` is found or a `NotFound` results. Otherwise, the returned `Item` may have a different `Id` than requested, such as the case where a duplicate `Id` was assigned to an `Item` and retained for compatibility.

Parameters **`item_id`** (`osid.id.Id`) – the `Id` of the `Item` to retrieve

Returns the returned `Item`

Return type `osid.assessment.Item`

Raise `NotFound` – no `Item` found with the given `Id`

Raise `NullArgument` – `item_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_items_by_ids`**(*item_ids*)

Gets an `ItemList` corresponding to the given `IdList`.

In plenary mode, the returned list contains all of the items specified in the `Id` list, in the order of the list, including duplicates, or an error results if an `Id` in the supplied list is not found or inaccessible. Otherwise, inaccessible `Items` may be omitted from the list and may present the elements in any order including returning a unique set.

Parameters **`item_ids`** (`osid.id.IdList`) – the list of `Ids` to retrieve

Returns the returned `Item` list

Return type `osid.assessment.ItemList`

Raise `NotFound` – an `Id was` not found

Raise `NullArgument` – `item_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_items_by_genus_type`**(*item_genus_type*)

Gets an `ItemList` corresponding to the given assessment item genus `Type` which does not include assessment items of genus types derived from the specified `Type`.

In plenary mode, the returned list contains all known assessment items or an error results. Otherwise, the returned list may contain only those assessment items that are accessible through this session.

Parameters **`item_genus_type`** (`osid.type.Type`) – an assessment item genus type

Returns the returned `Item` list

Return type `osid.assessment.ItemList`

Raise `NullArgument` – `item_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_items_by_parent_genus_type`**(*item_genus_type*)

Gets an `ItemList` corresponding to the given assessment item genus `Type` and include any additional assessment items with genus types derived from the specified `Type`.

In plenary mode, the returned list contains all known assessment items or an error results. Otherwise, the returned list may contain only those assessment items that are accessible through this session.

Parameters **`item_genus_type`** (`osid.type.Type`) – an assessment item genus type

Returns the returned `Item` list

Return type `osid.assessment.ItemList`

Raise `NullArgument` – `item_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_items_by_record_type`**(*item_record_type*)

Gets an `ItemList` containing the given assessment item record `Type`.

In plenary mode, the returned list contains all known items or an error results. Otherwise, the returned list may contain only those assessment items that are accessible through this session.

Parameters **`item_record_type`** (`osid.type.Type`) – an item record type

Returns the returned `Item` list

Return type `osid.assessment.ItemList`

Raise `NullArgument` – `item_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_items_by_question`**(*question_id*)

Gets an `ItemList` containing the given question.

In plenary mode, the returned list contains all known items or an error results. Otherwise, the returned list may contain only those assessment items that are accessible through this session.

Parameters **`question_id`** (`osid.id.Id`) – a question `Id`

Returns the returned `Item` list

Return type `osid.assessment.ItemList`

Raise `NullArgument` – `question_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_items_by_answer`**(*answer_id*)

Gets an `ItemList` containing the given answer.

In plenary mode, the returned list contains all known items or an error results. Otherwise, the returned list may contain only those assessment items that are accessible through this session.

Parameters **`answer_id`** (`osid.id.Id`) – an answer `Id`

Returns the returned `Item` list

Return type `osid.assessment.ItemList`

Raise `NullArgument` – `answer_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_items_by_learning_objective`**(*objective_id*)

Gets an `ItemList` containing the given learning objective.

In plenary mode, the returned list contains all known items or an error results. Otherwise, the returned list may contain only those assessment items that are accessible through this session.

Parameters **`objective_id`** (`osid.id.Id`) – a learning objective `Id`

Returns the returned `Item` list

Return type `osid.assessment.ItemList`

Raise `NullArgument` – `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_items_by_learning_objectives`**(*objective_ids*)

Gets an `ItemList` containing the given learning objectives.

In plenary mode, the returned list contains all known items or an error results. Otherwise, the returned list may contain only those assessment items that are accessible through this session.

Parameters **`objective_ids`** (`osid.id.IdList`) – a list of learning objective `Ids`

Returns the returned `Item` list

Return type `osid.assessment.ItemList`

Raise `NullArgument` – `objective_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`items`**

Gets all `Items`.

In plenary mode, the returned list contains all known items or an error results. Otherwise, the returned list may contain only those items that are accessible through this session.

Returns a list of `Items`

Return type `osid.assessment.ItemList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

Item Query Methods

`Bank.`**`bank_id`**

Gets the `Bank Id` associated with this session.

Returns the `Bank Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Bank.`**`bank`**

Gets the `Bank` associated with this session.

Returns the `Bank` associated with this session

Return type `osid.assessment.Bank`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`can_search_items`**`()`

Tests if this user can perform `Item` searches.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an pplication that may wish not to offer search operations to unauthorized users.

Returns `false` if search methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`use_federated_bank_view`**`()`

Federates the view for methods in this session.

A federated view will include assessments taken in banks which are children of this bank in the bank hierarchy.

compliance: mandatory – This method is must be implemented.

`Bank.`**`use_isolated_bank_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts searches to this bank only.

compliance: mandatory – This method is must be implemented.

`Bank.`**`item_query`**

Gets an assessment item query.

Returns the assessment item query

Return type `osid.assessment.ItemQuery`

compliance: mandatory – This method must be implemented.

`Bank.`**`get_items_by_query`**`(`*item_query*`)`

Gets a list of `Items` matching the given item query.

Parameters **`item_query`** (`osid.assessment.ItemQuery`) – the item query

Returns the returned `ItemList`

Return type `osid.assessment.ItemList`

Raise `NullArgument` – `item_query` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – `item_query` is not of this service

compliance: mandatory – This method must be implemented.

Item Search Methods

`Bank.`**`item_search`**

Gets an assessment item search.

Returns the assessment item search

Return type `osid.assessment.ItemSearch`

compliance: mandatory – This method must be implemented.

`Bank.`**`item_search_order`**

Gets an assessment item search order.

The `ItemSearchOrder` is supplied to an `ItemSearch` to specify the ordering of results.

Returns the assessment item search order

Return type `osid.assessment.ItemSearchOrder`

compliance: mandatory – This method must be implemented.

`Bank.`**`get_items_by_search`**(*item_query*, *item_search*)

Gets the search results matching the given search query using the given search.

Parameters

- **`item_query`** (`osid.assessment.ItemQuery`) – the item query
- **`item_search`** (`osid.assessment.ItemSearch`) – the item search

Returns the returned search results

Return type `osid.assessment.ItemSearchResults`

Raise `NullArgument` – `item_query` or `item_search` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – `item_search` or `item_query` is not of this service

compliance: mandatory – This method must be implemented.

`Bank.`**`get_item_query_from_inspector`**(*item_query_inspector*)

Gets an item query from an inspector.

The inspector is available from an `ItemSearchResults`.

Parameters **`item_query_inspector`** (`osid.assessment.ItemQueryInspector`) – a query inspector

Returns the item query

Return type `osid.assessment.ItemQuery`

Raise `NullArgument` – `item_query_inspector` is `null`

Raise `Unsupported` – `item_query_inspector` is not of this service

compliance: mandatory – This method must be implemented.

Item Admin Methods

`Bank.`**`bank_id`**

Gets the `Bank Id` associated with this session.

Returns the `Bank Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Bank.`**`bank`**

Gets the `Bank` associated with this session.

Returns the `Bank` associated with this session

Return type `osid.assessment.Bank`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`can_create_items`**`()`

Tests if this user can create `Items`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known creating an `Item` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer create operations to an unauthorized user.

Returns `false` if `Item` creation is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`can_create_item_with_record_types`**`(`*item_record_types*`)`

Tests if this user can create a single `Item` using the desired record types.

While `AssessmentManager.getItemRecordTypes()` can be used to examine which records are supported, this method tests which record(s) are required for creating a specific `Item`. Providing an empty array tests if an `Item` can be created with no records.

Parameters **`item_record_types`** (`osid.type.Type[]`) – array of item record types

Returns `true` if `Item` creation using the specified record `Types` is supported, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `item_record_types` is `null`

compliance: mandatory – This method must be implemented.

`Bank.`**`get_item_form_for_create`**`(`*item_record_types*`)`

Gets the assessment item form for creating new assessment items.

A new form should be requested for each create transaction.

Parameters **`item_record_types`** (`osid.type.Type[]`) – array of item record types to be included in the create operation or an empty list if none

Returns the assessment item form

Return type `osid.assessment.ItemForm`

Raise `NullArgument` – `item_record_types` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – unable to get form for requested record types

compliance: mandatory – This method must be implemented.

Bank.**create_item**(*item_form*)
Creates a new Item.

Parameters **item_form** (osid.assessment.ItemForm) – the form for this Item

Returns the new Item

Return type osid.assessment.Item

Raise IllegalState – item_form already used in a create transaction

Raise InvalidArgument – one or more of the form elements is invalid

Raise NullArgument – item_form is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure occurred

Raise Unsupported – item_form did not originate from get_item_form_for_create()

compliance: mandatory – This method must be implemented.

Bank.**can_update_items**()
Tests if this user can update Items.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating an Item will result in a PermissionDenied. This is intended as a hint to an application that may opt not to offer update operations to an unauthorized user.

Returns false if assessment item modification is not authorized, true otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

Bank.**get_item_form_for_update**(*item_id*)
Gets the assessment item form for updating an existing item.

A new item form should be requested for each update transaction.

Parameters **item_id** (osid.id.Id) – the Id of the Item

Returns the assessment item form

Return type osid.assessment.ItemForm

Raise NotFound – item_id is not found

Raise NullArgument – item_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure occurred

compliance: mandatory – This method must be implemented.

Bank.**update_item**(*item_form*)
Updates an existing item.

Parameters **item_form** (osid.assessment.ItemForm) – the form containing the elements to be updated

Raise IllegalState – item_form already used in an update transaction

Raise InvalidArgument – the form contains an invalid value

Raise NullArgument – item_form is null

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – `item_form` did not originate from `get_item_form_for_update()`

compliance: mandatory – This method must be implemented.

`Bank.`**`can_delete_items`**`()`

Tests if this user can delete `Items`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting an `Item` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer delete operations to an unauthorized user.

Returns `false` if `Item` deletion is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`delete_item`**`(`*item_id*`)`

Deletes the `Item` identified by the given `Id`.

Parameters **`item_id`** (`osid.id.Id`) – the `Id` of the `Item` to delete

Raise `NotFound` – an `Item` was not found identified by the given `Id`

Raise `NullArgument` – `item_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`can_manage_item_aliases`**`()`

Tests if this user can manage `Id` aliases for `Items`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known changing an alias will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer alias operations to an unauthorized user.

Returns `false` if `Item` aliasing is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`alias_item`**`(`*item_id*, *alias_id*`)`

Adds an `Id` to an `Item` for the purpose of creating compatibility.

The primary `Id` of the `Item` is determined by the provider. The new `Id` is an alias to the primary `Id`. If the alias is a pointer to another item, it is reassigned to the given item `Id`.

Parameters

- **`item_id`** (`osid.id.Id`) – the `Id` of an `Item`
- **`alias_id`** (`osid.id.Id`) – the alias `Id`

Raise `AlreadyExists` – `alias_id` is in use as a primary `Id`

Raise `NotFound` – `item_id` not found

Raise `NullArgument` – `item_id` or `alias_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`can_create_questions`**`()`

Tests if this user can create `Questions`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known creating a `Question` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer create operations to an unauthorized user.

Returns `false` if `Question` creation is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`can_create_question_with_record_types`**`(`*question_record_types*`)`

Tests if this user can create a single `Question` using the desired record types.

While `AssessmentManager.getQuestionRecordTypes()` can be used to examine which records are supported, this method tests which record(s) are required for creating a specific `Question`. Providing an empty array tests if a `Question` can be created with no records.

Parameters **`question_record_types`** (`osid.type.Type[]`) – array of question record types

Returns `true` if `Question` creation using the specified record `Types` is supported, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `question_record_types` is `null`

compliance: mandatory – This method must be implemented.

`Bank.`**`get_question_form_for_create`**`(`*item_id*, *question_record_types*`)`

Gets the question form for creating new questions.

A new form should be requested for each create transaction.

Parameters

- **`item_id`** (`osid.id.Id`) – an assessment item `Id`
- **`question_record_types`** (`osid.type.Type[]`) – array of question record types to be included in the create operation or an empty list if none

Returns the question form

Return type `osid.assessment.QuestionForm`

Raise `NullArgument` – `question_record_types` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – unable to get form for requested record types

compliance: mandatory – This method must be implemented.

`Bank.`**`create_question`**`(`*question_form*`)`

Creates a new `Question`.

Parameters **`question_form`** (`osid.assessment.QuestionForm`) – the form for this `Question`

Returns the new `Question`

Return type `osid.assessment.Question`

Raise `AlreadyExists` – a question already exists for this item

Raise `IllegalState` – `question_form` already used in a create transaction

Raise `InvalidArgument` – one or more of the form elements is invalid

Raise `NullArgument` – `question_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – `question_form` did not originate from `get_question_form_for_create()`

compliance: mandatory – This method must be implemented.

`Bank.`**`can_update_questions`**`()`

Tests if this user can update `Questions`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating a `Question` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer update operations to an unauthorized user.

Returns `false` if question modification is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`get_question_form_for_update`**`(`*question_id*`)`

Gets the question form for updating an existing question.

A new question form should be requested for each update transaction.

Parameters **`question_id`** (`osid.id.Id`) – the `Id` of the `Question`

Returns the question form

Return type `osid.assessment.QuestionForm`

Raise `NotFound` – `question_id` is not found

Raise `NullArgument` – `question_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`update_question`**`(`*question_form*`)`

Updates an existing question.

Parameters **`question_form`** (`osid.assessment.QuestionForm`) – the form containing the elements to be updated

Raise `IllegalState` – `question_form` already used in an update transaction

Raise `InvalidArgument` – the form contains an invalid value

Raise `NullArgument` – `question_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – `question_form` did not originate from `get_question_form_for_update()`

compliance: mandatory – This method must be implemented.

`Bank.`**`can_delete_questions`**`()`

Tests if this user can delete `Questions`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting a `Question` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer delete operations to an unauthorized user.

Returns `false` if `Question` deletion is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`delete_question`**`(`*question_id*`)`

Deletes the `Question` identified by the given `Id`.

Parameters **`question_id`** (`osid.id.Id`) – the `Id` of the `Question` to delete

Raise `NotFound` – a `Question` was not found identified by the given `Id`

Raise `NullArgument` – `question_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`can_create_answers`**`()`

Tests if this user can create `Answers`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known creating a `Answer` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer create operations to an unauthorized user.

Returns `false` if `Answer` creation is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`can_create_answers_with_record_types`**`(`*answer_record_types*`)`

Tests if this user can create a single `Answer` using the desired record types.

While `AssessmentManager.getAnswerRecordTypes()` can be used to examine which records are supported, this method tests which record(s) are required for creating a specific `Answer`. Providing an empty array tests if an `Answer` can be created with no records.

Parameters **`answer_record_types`** (`osid.type.Type[]`) – array of answer record types

Returns `true` if `Answer` creation using the specified record `Types` is supported, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `answern_record_types` is `null`

compliance: mandatory – This method must be implemented.

`Bank.`**`get_answer_form_for_create`**(*item_id*, *answer_record_types*)

Gets the answer form for creating new answers.

A new form should be requested for each create transaction.

Parameters

- **`item_id`** (`osid.id.Id`) – an assessment item `Id`
- **`answer_record_types`** (`osid.type.Type[]`) – array of answer record types to be included in the create operation or an empty list if none

Returns the answer form

Return type `osid.assessment.AnswerForm`

Raise `NullArgument` – `answer_record_types` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – unable to get form for requested record types

compliance: mandatory – This method must be implemented.

`Bank.`**`create_answer`**(*answer_form*)

Creates a new `Answer`.

Parameters **`answer_form`** (`osid.assessment.AnswerForm`) – the form for this `Answer`

Returns the new `Answer`

Return type `osid.assessment.Answer`

Raise `IllegalState` – `answer_form` already used in a create transaction

Raise `InvalidArgument` – one or more of the form elements is invalid

Raise `NullArgument` – `answer_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – `answer_form` did not originate from `get_answer_form_for_create()`

compliance: mandatory – This method must be implemented.

`Bank.`**`can_update_answers`**()

Tests if this user can update `Answers`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating an `Answer` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer update operations to an unauthorized user.

Returns `false` if answer modification is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

Bank.**get_answer_form_for_update**(*answer_id*)
Gets the answer form for updating an existing answer.

A new answer form should be requested for each update transaction.

Parameters **answer_id** (osid.id.Id) – the Id of the Answer

Returns the answer form

Return type osid.assessment.AnswerForm

Raise NotFound – answer_id is not found

Raise NullArgument – answer_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure occurred

compliance: mandatory – This method must be implemented.

Bank.**update_answer**(*answer_form*)
Updates an existing answer.

Parameters **answer_form** (osid.assessment.AnswerForm) – the form containing the elements to be updated

Raise IllegalState – answer_form already used in an update transaction

Raise InvalidArgument – the form contains an invalid value

Raise NullArgument – answer_form is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure occurred

Raise Unsupported – answer_form did not originate from get_answer_form_for_update()

compliance: mandatory – This method must be implemented.

Bank.**can_delete_answers**()
Tests if this user can delete Answers.

A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting an Answer will result in a PermissionDenied. This is intended as a hint to an application that may opt not to offer delete operations to an unauthorized user.

Returns false if Answer deletion is not authorized, true otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

Bank.**delete_answer**(*answer_id*)
Deletes the Answer identified by the given Id.

Parameters **answer_id** (osid.id.Id) – the Id of the Answer to delete

Raise NotFound – an Answer was not found identified by the given Id

Raise NullArgument – answer_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure occurred

compliance: mandatory – This method must be implemented.

Item Notification Methods

`Bank.`**`bank_id`**

Gets the `Bank Id` associated with this session.

Returns the `Bank Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Bank.`**`bank`**

Gets the `Bank` associated with this session.

Returns the `Bank` associated with this session

Return type `osid.assessment.Bank`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`can_register_for_item_notifications`**`()`

Tests if this user can register for `Item` notifications.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer notification operations.

Returns `false` if notification methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`use_federated_bank_view`**`()`

Federates the view for methods in this session.

A federated view will include assessments taken in banks which are children of this bank in the bank hierarchy.

compliance: mandatory – This method is must be implemented.

`Bank.`**`use_isolated_bank_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts searches to this bank only.

compliance: mandatory – This method is must be implemented.

`Bank.`**`reliable_item_notifications`**`()`

Reliable notifications are desired.

In reliable mode, notifications are to be acknowledged using `acknowledge_item_notification()` .

compliance: mandatory – This method is must be implemented.

`Bank.`**`unreliable_item_notifications`**`()`

Unreliable notifications are desired.

In unreliable mode, notifications do not need to be acknowledged.

compliance: mandatory – This method is must be implemented.

`Bank.`**`acknowledge_item_notification`**`(`*notification_id*`)`

Acknowledge an item notification.

Parameters **`notification_id`** (`osid.id.Id`) – the `Id` of the notification

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bank.`**`register_for_new_items`**`()`

Register for notifications of new assessment items.

`ItemReceiver.newItems()` is invoked when a new `Item` is created.

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`register_for_changed_items`**`()`

Registers for notification of updated assessment items.

`ItemReceiver.changedItems()` is invoked when an assessment item is changed.

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`register_for_changed_item`**`(`*item_id*`)`

Registers for notification of an updated assessment item.

`ItemReceiver.changedItems()` is invoked when the specified assessment item is changed.

Parameters **`item_id`** (`osid.id.Id`) – the `Id` of the `Assessment` to monitor

Raise `NotFound` – an `item` was not found identified by the given `Id`

Raise `NullArgument` – `item_id is null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`register_for_deleted_items`**`()`

Registers for notification of deleted assessment items.

`ItemReceiver.deletedItems()` is invoked when an assessment item is removed from the assessment bank.

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`register_for_deleted_item`**`(`*item_id*`)`

Registers for notification of a deleted assessment item.

`ItemReceiver.deletedItems()` is invoked when the specified assessment item is removed from the assessment bank.

Parameters **`item_id`** (`osid.id.Id`) – the `Id` of the `Item` to monitor

Raise `NotFound` – an `Item` was not found identified by the given `Id`

Raise `NullArgument` – `item_id is null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

Item Bank Methods

`Bank.`**`can_lookup_item_bank_mappings`**`()`

Tests if this user can perform lookups of item/bank mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known lookup methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if looking up mappings is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`use_comparative_bank_view`**`()`

The returns from the lookup methods may omit or translate elements based on this session, such as assessment, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`Bank.`**`use_plenary_bank_view`**`()`

A complete view of the `AssessmentTaken` and `Bank` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`Bank.`**`get_item_ids_by_bank`**`(`*bank_id*`)`

Gets the list of `Item Ids` associated with a `Bank`.

Parameters **`bank_id`** (`osid.id.Id`) – `Id` of the `Bank`

Returns list of related item `Ids`

Return type `osid.id.IdList`

Raise `NotFound` – `bank_id` is not found

Raise `NullArgument` – `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_items_by_bank`**`(`*bank_id*`)`

Gets the list of `Items` associated with a `Bank`.

Parameters **`bank_id`** (`osid.id.Id`) – `Id` of the `Bank`

Returns list of related items

Return type `osid.assessment.ItemList`

Raise `NotFound` – `bank_id` is not found

Raise `NullArgument` – `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_item_ids_by_banks`**(*bank_ids*)
Gets the list of `Item Ids` corresponding to a list of `Banks`.

Parameters **`bank_ids`** (`osid.id.IdList`) – list of bank `Ids`

Returns list of bank `Ids`

Return type `osid.id.IdList`

Raise `NullArgument` – `bank_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – assessment failure

compliance: mandatory – This method must be implemented.

`Bank.`**`get_items_by_banks`**(*bank_ids*)
Gets the list of `Items` corresponding to a list of `Banks`.

Parameters **`bank_ids`** (`osid.id.IdList`) – list of bank `Ids`

Returns list of items

Return type `osid.assessment.ItemList`

Raise `NullArgument` – `bank_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – assessment failure

compliance: mandatory – This method must be implemented.

`Bank.`**`get_bank_ids_by_item`**(*item_id*)
Gets the list of `Bank Ids` mapped to an `Item`.

Parameters **`item_id`** (`osid.id.Id`) – `Id` of an `Item`

Returns list of bank `Ids`

Return type `osid.id.IdList`

Raise `NotFound` – `item_id` is not found

Raise `NullArgument` – `item_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – assessment failure

compliance: mandatory – This method must be implemented.

`Bank.`**`get_banks_by_item`**(*item_id*)
Gets the list of `Banks` mapped to an `Item`.

Parameters **`item_id`** (`osid.id.Id`) – `Id` of an `Item`

Returns list of banks

Return type `osid.assessment.BankList`

Raise `NotFound` – `item_id` is not found

Raise `NullArgument` – `item_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – assessment failure

compliance: mandatory – This method must be implemented.

Item Bank Assignment Methods

`Bank.`**`can_assign_items`**`()`

Tests if this user can alter item/bank mappings.

A return of true does not guarantee successful assessment. A return of false indicates that it is known mapping methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer assignment operations to unauthorized users.

Returns `false` if mapping is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`can_assign_items_to_bank`**`(`*bank_id*`)`

Tests if this user can alter item/bank mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known mapping methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Parameters **`bank_id`** (`osid.id.Id`) – the `Id` of the `Bank`

Returns `false` if mapping is not authorized, `true` otherwise

Return type `boolean`

Raise `NullArgument` – `bank_id` is `null`

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assignable_bank_ids`**`(`*bank_id*`)`

Gets a list of banks including and under the given banks node in which any assessment taken can be assigned.

Parameters **`bank_id`** (`osid.id.Id`) – the `Id` of the `Bank`

Returns list of assignable bank `Ids`

Return type `osid.id.IdList`

Raise `NullArgument` – `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assignable_bank_ids_for_item`**`(`*bank_id*, *item_id*`)`

Gets a list of banks including and under the given bank node in which a specific item can be assigned.

Parameters

- **bank_id** (osid.id.Id) – the Id of the Bank
- **item_id** (osid.id.Id) – the Id of the Item

Returns list of assignable bank Ids

Return type osid.id.IdList

Raise NullArgument – bank_id or item_id is null

Raise OperationFailed – unable to complete request

compliance: mandatory – This method must be implemented.

Bank.**assign_item_to_bank**(*item_id*, *bank_id*)

Adds an existing Item to a Bank.

Parameters

- **item_id** (osid.id.Id) – the Id of the Item
- **bank_id** (osid.id.Id) – the Id of the Bank

Raise AlreadyExists – item_id is already assigned to bank_id

Raise NotFound – item_id or bank_id not found

Raise NullArgument – item_id or bank_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure occurred

compliance: mandatory – This method must be implemented.

Bank.**unassign_item_from_bank**(*item_id*, *bank_id*)

Removes an Item from a Bank.

Parameters

- **item_id** (osid.id.Id) – the Id of the Item
- **bank_id** (osid.id.Id) – the Id of the Bank

Raise NotFound – item_id or bank_id not found or item_id not assigned to bank_id

Raise NullArgument – item_id or bank_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure occurred

compliance: mandatory – This method must be implemented.

Bank.**reassign_item_to_billing**(*item_id*, *from_bank_id*, *to_bank_id*)

Moves an Item from one Bank to another.

Mappings to other Banks are unaffected.

Parameters

- **item_id** (osid.id.Id) – the Id of the Item
- **from_bank_id** (osid.id.Id) – the Id of the current Bank
- **to_bank_id** (osid.id.Id) – the Id of the destination Bank

Raise NotFound – item_id, from_bank_id, or to_bank_id not found or item_id not mapped to from_bank_id

Raise `NullArgument` – `item_id, from_bank_id`, or `to_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Assessment Lookup Methods

`Bank.`**`bank_id`**

Gets the `Bank Id` associated with this session.

Returns the `Bank Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Bank.`**`bank`**

Gets the `Bank` associated with this session.

Returns the `Bank` associated with this session

Return type `osid.assessment.Bank`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`can_lookup_assessments`**`()`

Tests if this user can perform `Assessment` lookups.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if lookup methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`use_comparative_assessment_view`**`()`

The returns from the lookup methods may omit or translate elements based on this session, such as assessment, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`Bank.`**`use_plenary_assessment_view`**`()`

A complete view of the `Assessment` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`Bank.`**`use_federated_bank_view`**`()`

Federates the view for methods in this session.

A federated view will include assessments taken in banks which are children of this bank in the bank hierarchy.

compliance: mandatory – This method is must be implemented.

`Bank.`**`use_isolated_bank_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts searches to this bank only.

compliance: mandatory – This method is must be implemented.

`Bank.`**`get_assessment`**`(`*assessment_id*`)`

Gets the `Assessment` specified by its `Id`.

In plenary mode, the exact `Id` is found or a `NotFound` results. Otherwise, the returned `Assessment` may have a different `Id` than requested, such as the case where a duplicate `Id` was assigned to a `Assessment` and retained for compatibility.

Parameters **`assessment_id`** (`osid.id.Id`) – `Id` of the `Assessment`

Returns the assessment

Return type `osid.assessment.Assessment`

Raise `NotFound` – `assessment_id` not found

Raise `NullArgument` – `assessment_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method is must be implemented.

`Bank.`**`get_assessments_by_ids`**`(`*assessment_ids*`)`

Gets an `AssessmentList` corresponding to the given `IdList`.

In plenary mode, the returned list contains all of the assessments specified in the `Id` list, in the order of the list, including duplicates, or an error results if an `Id` in the supplied list is not found or inaccessible. Otherwise, inaccessible `Assessments` may be omitted from the list and may present the elements in any order including returning a unique set.

Parameters **`assessment_ids`** (`osid.id.IdList`) – the list of `Ids` to retrieve

Returns the returned `Assessment` list

Return type `osid.assessment.AssessmentList`

Raise `NotFound` – an `Id was` not found

Raise `NullArgument` – `assessment_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – assessment failure

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_by_genus_type`**`(`*assessment_genus_type*`)`

Gets an `AssessmentList` corresponding to the given assessment genus `Type` which does not include assessments of types derived from the specified `Type`.

In plenary mode, the returned list contains all known assessments or an error results. Otherwise, the returned list may contain only those assessments that are accessible through this session.

Parameters **`assessment_genus_type`** (`osid.type.Type`) – an assessment genus type

Returns the returned `Assessment` list

Return type `osid.assessment.AssessmentList`

Raise `NullArgument` – `assessment_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_by_parent_genus_type`**(*assessment_genus_type*)

Gets an `AssessmentList` corresponding to the given assessment genus `Type` and include any additional assessments with genus types derived from the specified `Type`.

In plenary mode, the returned list contains all known assessments or an error results. Otherwise, the returned list may contain only those assessments that are accessible through this session.

Parameters **`assessment_genus_type`** (`osid.type.Type`) – an assessment genus type

Returns the returned `Assessment` list

Return type `osid.assessment.AssessmentList`

Raise `NullArgument` – `assessment_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_by_record_type`**(*assessment_record_type*)

Gets an `AssessmentList` corresponding to the given assessment record `Type`.

The set of assessments implementing the given record type is returned. In plenary mode, the returned list contains all known assessments or an error results. Otherwise, the returned list may contain only those assessments that are accessible through this session.

Parameters **`assessment_record_type`** (`osid.type.Type`) – an assessment record type

Returns the returned `Assessment` list

Return type `osid.assessment.AssessmentList`

Raise `NullArgument` – `assessment_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`assessments`**

Gets all `Assessments`.

In plenary mode, the returned list contains all known assessments or an error results. Otherwise, the returned list may contain only those assessments that are accessible through this session.

Returns a list of `Assessments`

Return type `osid.assessment.AssessmentList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

Assessment Query Methods

`Bank.`**`bank_id`**

Gets the `Bank Id` associated with this session.

Returns the `Bank Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Bank.`**`bank`**

Gets the `Bank` associated with this session.

Returns the `Bank` associated with this session

Return type `osid.assessment.Bank`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`can_search_assessments`**`()`

Tests if this user can perform `Assessment` searches.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an pplication that may wish not to offer search operations to unauthorized users.

Returns `false` if search methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`use_federated_bank_view`**`()`

Federates the view for methods in this session.

A federated view will include assessments taken in banks which are children of this bank in the bank hierarchy.

compliance: mandatory – This method is must be implemented.

`Bank.`**`use_isolated_bank_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts searches to this bank only.

compliance: mandatory – This method is must be implemented.

`Bank.`**`assessment_query`**

Gets an assessment query.

Returns the assessment query

Return type `osid.assessment.AssessmentQuery`

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_by_query`**`(`*assessment_query*`)`

Gets a list of `Assessments` matching the given assessment query.

Parameters **`assessment_query`** (`osid.assessment.AssessmentQuery`) – the assessment query

Returns the returned `AssessmentList`

Return type `osid.assessment.AssessmentList`

Raise `NullArgument` – `assessment_query` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – `assessment_query` is not of this service

compliance: mandatory – This method must be implemented.

Assessment Admin Methods

`Bank.`**`bank_id`**

Gets the `Bank Id` associated with this session.

Returns the `Bank Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Bank.`**`bank`**

Gets the `Bank` associated with this session.

Returns the `Bank` associated with this session

Return type `osid.assessment.Bank`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`can_create_assessments`**`()`

Tests if this user can create `Assessments`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known creating an `Assessment` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer create operations to an unauthorized user.

Returns `false` if `Assessment` creation is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`can_create_assessment_with_record_types`**`(`*assessment_record_types*`)`

Tests if this user can create a single `Assessment` using the desired record interface types.

While `AssessmentManager.getAssessmentRecordTypes()` can be used to examine which record interfaces are supported, this method tests which record(s) are required for creating a specific `Assessment`. Providing an empty array tests if an `Assessment` can be created with no records.

Parameters **`assessment_record_types`** (`osid.type.Type[]`) – array of assessment record types

Returns `true` if `Assessment` creation using the specified record `Types` is supported, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `assessment_record_types` is `null`

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessment_form_for_create`**(*assessment_record_types*)

Gets the assessment form for creating new assessments.

A new form should be requested for each create transaction.

Parameters **`assessment_record_types`** (`osid.type.Type[]`) – array of assessment record types to be included in the create operation or an empty list if none

Returns the assessment form

Return type `osid.assessment.AssessmentForm`

Raise `NullArgument` – `assessment_record_types` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – unable to get form for requested record types

compliance: mandatory – This method must be implemented.

`Bank.`**`create_assessment`**(*assessment_form*)

Creates a new `Assessment`.

Parameters **`assessment_form`** (`osid.assessment.AssessmentForm`) – the form for this `Assessment`

Returns the new `Assessment`

Return type `osid.assessment.Assessment`

Raise `IllegalState` – `assessment_form` already used in a create transaction

Raise `InvalidArgument` – one or more of the form elements is invalid

Raise `NullArgument` – `assessment_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – `assessment_form` did not originate from `get_assessment_form_for_create()`

compliance: mandatory – This method must be implemented.

`Bank.`**`can_update_assessments`**()

Tests if this user can update `Assessments`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating an `Assessment` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer update operations to an unauthorized user.

Returns `false` if `Assessment` modification is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

Bank.**get_assessment_form_for_update**(*assessment_id*)

Gets the assessment form for updating an existing assessment.

A new assessment form should be requested for each update transaction.

Parameters **assessment_id** (osid.id.Id) – the Id of the Assessment

Returns the assessment form

Return type osid.assessment.AssessmentForm

Raise NotFound – assessment_id is not found

Raise NullArgument – assessment_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure occurred

compliance: mandatory – This method must be implemented.

Bank.**update_assessment**(*assessment_form*)

Updates an existing assessment.

Parameters **assessment_form** (osid.assessment.AssessmentForm) – the form containing the elements to be updated

Raise IllegalState – assessment_form already used in an update transaction

Raise InvalidArgument – the form contains an invalid value

Raise NullArgument – assessment_form is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure occurred

Raise Unsupported – assessment_form did not originate from get_assessment_form_for_update()

compliance: mandatory – This method must be implemented.

Bank.**can_delete_assessments**()

Tests if this user can delete Assessments.

A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting an Assessment will result in a PermissionDenied. This is intended as a hint to an application that may opt not to offer delete operations to an unauthorized user.

Returns false if Assessment deletion is not authorized, true otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

Bank.**delete_assessment**(*assessment_id*)

Deletes an Assessment.

Parameters **assessment_id** (osid.id.Id) – the Id of the Assessment to remove

Raise NotFound – assessment_id not found

Raise NullArgument – assessment_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`can_manage_assessment_aliases`**`()`

Tests if this user can manage `Id` aliases for `Assessments`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known changing an alias will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer alias operations to an unauthorized user.

Returns `false` if `Assessment` aliasing is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`alias_assessment`**`(`*assessment_id*, *alias_id*`)`

Adds an `Id` to an `Assessment` for the purpose of creating compatibility.

The primary `Id` of the `Assessment` is determined by the provider. The new `Id` is an alias to the primary `Id`. If the alias is a pointer to another assessment, it is reassigned to the given assessment `Id`.

Parameters

- **`assessment_id`** (`osid.id.Id`) – the `Id` of an `Assessment`
- **`alias_id`** (`osid.id.Id`) – the alias `Id`

Raise `AlreadyExists` – `alias_id` is in use as a primary `Id`

Raise `NotFound` – `assessment_id` not found

Raise `NullArgument` – `assessment_id` or `alias_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

Assessment Bank Methods

`Bank.`**`can_lookup_assessment_bank_mappings`**`()`

Tests if this user can perform lookups of assessment/bank mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known lookup methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if looking up mappings is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`use_comparative_bank_view`**`()`

The returns from the lookup methods may omit or translate elements based on this session, such as assessment, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`Bank.`**`use_plenary_bank_view`**`()`

A complete view of the `AssessmentTaken` and `Bank` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`Bank.`**`get_assessment_ids_by_bank`**`(`*bank_id*`)`

Gets the list of `Assessment Ids` associated with a `Bank`.

Parameters **`bank_id`** (`osid.id.Id`) – `Id` of the `Bank`

Returns list of related assessment `Ids`

Return type `osid.id.IdList`

Raise `NotFound` – `bank_id` is not found

Raise `NullArgument` – `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_by_bank`**`(`*bank_id*`)`

Gets the list of `Assessments` associated with a `Bank`.

Parameters **`bank_id`** (`osid.id.Id`) – `Id` of the `Bank`

Returns list of related assessments

Return type `osid.assessment.AssessmentList`

Raise `NotFound` – `bank_id` is not found

Raise `NullArgument` – `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessment_ids_by_banks`**`(`*bank_ids*`)`

Gets the list of `Assessment Ids` corresponding to a list of `Banks`.

Parameters **`bank_ids`** (`osid.id.IdList`) – list of bank `Ids`

Returns list of bank `Ids`

Return type `osid.id.IdList`

Raise `NullArgument` – `bank_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_by_banks`**`(`*bank_ids*`)`

Gets the list of `Assessments` corresponding to a list of `Banks`.

Parameters **`bank_ids`** (`osid.id.IdList`) – list of bank `Ids`

Returns list of assessments

Return type `osid.assessment.AssessmentList`

Raise `NullArgument` – `bank_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_bank_ids_by_assessment`**`(`*assessment_id*`)`

Gets the list of `Bank Ids` mapped to an `Assessment`.

Parameters **`assessment_id`** (`osid.id.Id`) – `Id` of an `Assessment`

Returns list of bank `Ids`

Return type `osid.id.IdList`

Raise `NotFound` – `assessment_id` is not found

Raise `NullArgument` – `assessment_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_banks_by_assessment`**`(`*assessment_id*`)`

Gets the list of `Banks` mapped to an `Assessment`.

Parameters **`assessment_id`** (`osid.id.Id`) – `Id` of an `Assessment`

Returns list of banks

Return type `osid.assessment.BankList`

Raise `NotFound` – `assessment_id` is not found

Raise `NullArgument` – `assessment_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

Assessment Bank Assignment Methods

`Bank.`**`can_assign_assessments`**`()`

Tests if this user can alter assessment/bank mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known mapping methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if mapping is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`can_assign_assessments_to_bank`**`(`*bank_id*`)`

Tests if this user can alter assessment/bank mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known mapping methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Parameters **bank_id** (`osid.id.Id`) – the `Id` of the `Bank`

Returns `false` if mapping is not authorized, `true` otherwise

Return type `boolean`

Raise `NullArgument` – `bank_id` is `null`

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assignable_bank_ids`**(*bank_id*)

Gets a list of banks including and under the given banks node in which any assessment taken can be assigned.

Parameters **bank_id** (`osid.id.Id`) – the `Id` of the `Bank`

Returns list of assignable bank `Ids`

Return type `osid.id.IdList`

Raise `NullArgument` – `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assignable_bank_ids_for_assessment`**(*bank_id*, *assessment_id*)

Gets a list of bank including and under the given bank node in which a specific assessment can be assigned.

Parameters

- **bank_id** (`osid.id.Id`) – the `Id` of the `Bank`
- **assessment_id** (`osid.id.Id`) – the `Id` of the `Assessment`

Returns list of assignable bank `Ids`

Return type `osid.id.IdList`

Raise `NullArgument` – `bank_id` or `assessment_id` is `null`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`Bank.`**`assign_assessment_to_bank`**(*assessment_id*, *bank_id*)

Adds an existing `Assessment` to a `Bank`.

Parameters

- **assessment_id** (`osid.id.Id`) – the `Id` of the `Assessment`
- **bank_id** (`osid.id.Id`) – the `Id` of the `Bank`

Raise `AlreadyExists` – `assessment_id` is already assigned to `bank_id`

Raise `NotFound` – `assessment_id` or `bank_id` not found

Raise `NullArgument` – `assessment_id` or `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

Bank.**unassign_assessment_from_bank**(*assessment_id*, *bank_id*)

Removes an `Assessment` from a `Bank`.

Parameters

- **assessment_id** (`osid.id.Id`) – the `Id` of the `Assessment`
- **bank_id** (`osid.id.Id`) – the `Id` of the `Bank`

Raise `NotFound` – `assessment_id` or `bank_id` not found or `assessment_id` not assigned to `bank_id`

Raise `NullArgument` – `assessment_id` or `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

Bank.**reassign_assessment_to_billing**(*assessment_id*, *from_bank_id*, *to_bank_id*)

Moves an `Assessment` from one `Bank` to another.

Mappings to other `Banks` are unaffected.

Parameters

- **assessment_id** (`osid.id.Id`) – the `Id` of the `Assessment`
- **from_bank_id** (`osid.id.Id`) – the `Id` of the current `Bank`
- **to_bank_id** (`osid.id.Id`) – the `Id` of the destination `Bank`

Raise `NotFound` – `assessment_id, from_bank_id,` or `to_bank_id` not found or `assessment_id` not mapped to `from_bank_id`

Raise `NullArgument` – `assessment_id, from_bank_id,` or `to_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Assessment Basic Authoring Methods

Bank.**bank_id**

Gets the `Bank Id` associated with this session.

Returns the `Bank Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

Bank.**bank**

Gets the `Bank` associated with this session.

Returns the `Bank` associated with this session

Return type `osid.assessment.Bank`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`can_author_assessments`**`()`

Tests if this user can author assessments.

A return of true does not guarantee successful authorization. A return of false indicates that it is known mapping methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer authoring operations to unauthorized users.

Returns `false` if mapping is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`get_items`**`()`

Gets all `Items`.

In plenary mode, the returned list contains all known items or an error results. Otherwise, the returned list may contain only those items that are accessible through this session.

Returns a list of `Items`

Return type `osid.assessment.ItemList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`add_item`**`(`*assessment_id*, *item_id*`)`

Adds an existing `Item` to an assessment.

Parameters

- **`assessment_id`** (`osid.id.Id`) – the `Id` of the `Assessment`
- **`item_id`** (`osid.id.Id`) – the `Id` of the `Item`

Raise `NotFound` – `assessment_id` or `item_id` not found

Raise `NullArgument` – `assessment_id` or `item_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`remove_item`**`(`*assessment_id*, *item_id*`)`

Removes an `Item` from this assessment.

Parameters

- **`assessment_id`** (`osid.id.Id`) – the `Id` of the `Assessment`
- **`item_id`** (`osid.id.Id`) – the `Id` of the `Item`

Raise `NotFound` – `assessment_id` or `item_id` not found or `item_id` not on `assessmentid`

Raise `NullArgument` – `assessment_id` or `item_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`move_item`**(*assessment_id*, *item_id*, *preceeding_item_id*)

Moves an existing item to follow another item in an assessment.

Parameters

- **`assessment_id`** (`osid.id.Id`) – the `Id` of the `Assessment`
- **`item_id`** (`osid.id.Id`) – the `Id` of an `Item`
- **`preceeding_item_id`** (`osid.id.Id`) – the `Id` of a preceeding `Item` in the sequence

Raise `NotFound` – `assessment_id` is not found, or `item_id` or `preceeding_item_id` not on `assessment_id`

Raise `NullArgument` – `assessment_id, item_id` or `preceeding_item_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`order_items`**(*item_ids*, *assessment_id*)

Sequences existing items in an assessment.

Parameters

- **`item_ids`** (`osid.id.Id[]`) – the `Id` of the `Items`
- **`assessment_id`** (`osid.id.Id`) – the `Id` of the `Assessment`

Raise `NotFound` – `assessment_id` is not found or an `item_id` is not on `assessment_id`

Raise `NullArgument` – `assessment_id` or `item_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

Assessment Offered Lookup Methods

`Bank.`**`bank_id`**

Gets the `Bank Id` associated with this session.

Returns the `Bank Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Bank.`**`bank`**

Gets the `Bank` associated with this session.

Returns the `Bank` associated with this session

Return type `osid.assessment.Bank`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

Bank.**can_lookup_assessments_offered**()

Tests if this user can perform `AssessmentOffered` lookups.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if lookup methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

Bank.**use_comparative_assessment_offered_view**()

The returns from the lookup methods may omit or translate elements based on this session, such as assessment, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

Bank.**use_plenary_assessment_offered_view**()

A complete view of the `AssessmentOffered` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

Bank.**use_federated_bank_view**()

Federates the view for methods in this session.

A federated view will include assessments taken in banks which are children of this bank in the bank hierarchy.

compliance: mandatory – This method is must be implemented.

Bank.**use_isolated_bank_view**()

Isolates the view for methods in this session.

An isolated view restricts searches to this bank only.

compliance: mandatory – This method is must be implemented.

Bank.**get_assessment_offered**(*assessment_offered_id*)

Gets the `AssessmentOffered` specified by its `Id`.

In plenary mode, the exact `Id` is found or a `NotFound` results. Otherwise, the returned `AssessmentOffered` may have a different `Id` than requested, such as the case where a duplicate `Id` was assigned to an `AssessmentOffered` and retained for compatibility.

Parameters **assessment_offered_id** (`osid.id.Id`) – `Id` of the `AssessmentOffered`

Returns the assessment offered

Return type `osid.assessment.AssessmentOffered`

Raise `NotFound` – `assessment_offered_id` not found

Raise `NullArgument` – `assessment_offered_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method is must be implemented.

`Bank.`**`get_assessments_offered_by_ids`**(*assessment_offered_ids*)

Gets an `AssessmentOfferedList` corresponding to the given `IdList`.

In plenary mode, the returned list contains all of the assessments specified in the `Id` list, in the order of the list, including duplicates, or an error results if an `Id` in the supplied list is not found or inaccessible. Otherwise, inaccessible `AssessmentOffered` objects may be omitted from the list and may present the elements in any order including returning a unique set.

Parameters **`assessment_offered_ids`** (`osid.id.IdList`) – the list of `Ids` to retrieve

Returns the returned `AssessmentOffered` list

Return type `osid.assessment.AssessmentOfferedList`

Raise `NotFound` – an `Id was` not found

Raise `NullArgument` – `assessment_offered_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – assessment failure

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_offered_by_genus_type`**(*assessment_offered_genus_type*)

Gets an `AssessmentOfferedList` corresponding to the given assessment offered genus `Type` which does not include assessments of types derived from the specified `Type`.

In plenary mode, the returned list contains all known assessments offered or an error results. Otherwise, the returned list may contain only those assessments offered that are accessible through this session.

Parameters **`assessment_offered_genus_type`** (`osid.type.Type`) – an assessment offered genus type

Returns the returned `AssessmentOffered` list

Return type `osid.assessment.AssessmentOfferedList`

Raise `NullArgument` – `assessment_offered_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_offered_by_parent_genus_type`**(*assessment_offered_genus_type*)

Gets an `AssessmentOfferedList` corresponding to the given assessment offered genus `Type` and include any additional assessments with genus types derived from the specified `Type`.

In plenary mode, the returned list contains all known assessments or an error results. Otherwise, the returned list may contain only those assessments offered that are accessible through this session.

Parameters **`assessment_offered_genus_type`** (`osid.type.Type`) – an assessment offered genus type

Returns the returned `AssessmentOffered` list

Return type `osid.assessment.AssessmentOfferedList`

Raise `NullArgument` – `assessment_offered_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_offered_by_record_type`**(*assessment_record_type*)

Gets an `AssessmentOfferedList` corresponding to the given assessment offered record `Type`.

The set of assessments implementing the given record type is returned. In plenary mode, the returned list contains all known assessments offered or an error results. Otherwise, the returned list may contain only those assessments offered that are accessible through this session.

Parameters **`assessment_record_type`** (`osid.type.Type`) – an assessment offered record type

Returns the returned `AssessmentOffered` list

Return type `osid.assessment.AssessmentOfferedList`

Raise `NullArgument` – `assessment_offered_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_offered_by_date`**(*start*, *end*)

Gets an `AssessmentOfferedList` that have designated start times where the start times fall in the given range inclusive.

In plenary mode, the returned list contains all known assessments offered or an error results. Otherwise, the returned list may contain only those assessments offered that are accessible through this session.

Parameters

- **`start`** (`osid.calendaring.DateTime`) – start of time range
- **`end`** (`osid.calendaring.DateTime`) – end of time range

Returns the returned `AssessmentOffered` list

Return type `osid.assessment.AssessmentOfferedList`

Raise `InvalidArgument` – `end` is less than `start`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_offered_for_assessment`**(*assessment_id*)

Gets an `AssessmentOfferedList` by the given assessment.

In plenary mode, the returned list contains all known assessments offered or an error results. Otherwise, the returned list may contain only those assessments offered that are accessible through this session.

Parameters **`assessment_id`** (`osid.id.Id`) – `Id` of an `Assessment`

Returns the returned `AssessmentOffered` list

Return type `osid.assessment.AssessmentOfferedList`

Raise `NullArgument` – `assessment_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`assessments_offered`**

Gets all `AssessmentOffered` elements.

In plenary mode, the returned list contains all known assessments offered or an error results. Otherwise, the returned list may contain only those assessments offered that are accessible through this session.

Returns a list of `AssessmentOffered` elements

Return type `osid.assessment.AssessmentOfferedList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

Assessment Offered Query Methods

`Bank.`**`bank_id`**

Gets the `Bank Id` associated with this session.

Returns the `Bank Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Bank.`**`bank`**

Gets the `Bank` associated with this session.

Returns the `Bank` associated with this session

Return type `osid.assessment.Bank`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`can_search_assessments_offered`**`()`

Tests if this user can perform `AssessmentOffered` searches.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may wish not to offer search operations to unauthorized users.

Returns `false` if search methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`use_federated_bank_view`**`()`

Federates the view for methods in this session.

A federated view will include assessments taken in banks which are children of this bank in the bank hierarchy.

compliance: mandatory – This method is must be implemented.

`Bank.`**`use_isolated_bank_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts searches to this bank only.

compliance: mandatory – This method is must be implemented.

`Bank.`**`assessment_offered_query`**

Gets an assessment offered query.

Returns the assessment offered query

Return type `osid.assessment.AssessmentOfferedQuery`

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_offered_by_query`**`(`*assessment_offered_query*`)`

Gets a list of `AssessmentOffered` elements matching the given assessment offered query.

Parameters **`assessment_offered_query`** (`osid.assessment.AssessmentOfferedQuery`) – the assessment offered query

Returns the returned `AssessmentOfferedList`

Return type `osid.assessment.AssessmentOfferedList`

Raise `NullArgument` – `assessment_offered_query` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – `assessment_offered_query` is not of this service

compliance: mandatory – This method must be implemented.

Assessment Offered Admin Methods

`Bank.`**`bank_id`**

Gets the `Bank Id` associated with this session.

Returns the `Bank Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Bank.`**`bank`**

Gets the `Bank` associated with this session.

Returns the `Bank` associated with this session

Return type `osid.assessment.Bank`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`can_create_assessments_offered`**`()`

Tests if this user can create `AssessmentOffered` objects.

A return of true does not guarantee successful authoriization. A return of false indicates that it is known creating an `AssessmentOffered` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer create operations to an unauthorized user.

Returns `false` if `AssessmentOffered` creation is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`can_create_assessment_offered_with_record_types`**(*assessment_offered_record_types*)

Tests if this user can create a single `AssessmentOffered` using the desired record types.

While `AssessmentManager.getAssessmentOfferedRecordTypes()` can be used to examine which records are supported, this method tests which record(s) are required for creating a specific `AssessmentOffered`. Providing an empty array tests if an `AssessmentOffered` can be created with no records.

Parameters **`assessment_offered_record_types`** (`osid.type.Type[]`) – array of assessment offered record types

Returns `true` if `AssessmentOffered` creation using the specified record `Types` is supported, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `assessment_offered_record_types` is `null`

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessment_offered_form_for_create`**(*assessment_id*, *assessment_offered_record_types*)

Gets the assessment offered form for creating new assessments offered.

A new form should be requested for each create transaction.

Parameters

- **`assessment_id`** (`osid.id.Id`) – the `Id` of the related `Assessment`
- **`assessment_offered_record_types`** (`osid.type.Type[]`) – array of assessment offered record types to be included in the create operation or an empty list if none

Returns the assessment offered form

Return type `osid.assessment.AssessmentOfferedForm`

Raise `NotFound` – `assessment_id` is not found

Raise `NullArgument` – `assessment_id` or `assessment_offered_record_types` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – unable to get form for requested record types

compliance: mandatory – This method must be implemented.

`Bank.`**`create_assessment_offered`**(*assessment_offered_form*)

Creates a new `AssessmentOffered`.

Parameters **`assessment_offered_form`** (`osid.assessment.AssessmentOfferedForm`) – the form for this `AssessmentOffered`

Returns the new `AssessmentOffered`

Return type `osid.assessment.AssessmentOffered`

Raise `IllegalState` – `assessment_offrered_form` already used in a create transaction

Raise `InvalidArgument` – one or more of the form elements is invalid

Raise `NullArgument` – `assessment_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – `assessment_form` did not originate from `get_assessment_form_for_create()`

compliance: mandatory – This method must be implemented.

`Bank.`**`can_update_assessments_offered`**`()`

Tests if this user can update `AssessmentOffered` objects.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating an `AssessmentOffered` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer update operations to an unauthorized user.

Returns `false` if `Assessment` modification is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessment_offered_form_for_update`**`(`*assessment_offered_id*`)`

Gets the assessment offered form for updating an existing assessment offered.

A new assessment offered form should be requested for each update transaction.

Parameters **`assessment_offered_id`** (`osid.id.Id`) – the `Id` of the `AssessmentOffered`

Returns the assessment offered form

Return type `osid.assessment.AssessmentOfferedForm`

Raise `NotFound` – `assessment_offered_id` is not found

Raise `NullArgument` – `assessment_offered_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`update_assessment_offered`**`(`*assessment_offered_form*`)`

Updates an existing assessment offered.

Parameters **`assessment_offered_form`** (`osid.assessment.AssessmentOfferedForm`) – the form containing the elements to be updated

Raise `IllegalState` – `assessment_offrered_form` already used in an update transaction

Raise `InvalidArgument` – the form contains an invalid value

Raise `NullArgument` – `assessment_offered_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – `assessment_form` did not originate from `get_assessment_form_for_update()`

compliance: mandatory – This method must be implemented.

`Bank.`**`can_delete_assessments_offered`**`()`

Tests if this user can delete `AssessmentsOffered`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting an `AssessmentOffered` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer a delete operations to unauthorized users.

Returns `false` if `AssessmentOffered` deletion is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`delete_assessment_offered`**`(`*assessment_offered_id*`)`

Deletes an `AssessmentOffered`.

Parameters **`assessment_offered_id`** (`osid.id.Id`) – the `Id` of the `AssessmentOffered` to remove

Raise `NotFound` – `assessment_offered_id` not found

Raise `NullArgument` – `assessment_offered_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`can_manage_assessment_offered_aliases`**`()`

Tests if this user can manage `Id` aliases for `AssessmentsOffered`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known changing an alias will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer alias operations to an unauthorized user.

Returns `false` if `AssessmentOffered` aliasing is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`alias_assessment_offered`**`(`*assessment_offered_id*, *alias_id*`)`

Adds an `Id` to an `AssessmentOffered` for the purpose of creating compatibility.

The primary `Id` of the `AssessmentOffered` is determined by the provider. The new `Id` is an alias to the primary `Id`. If the alias is a pointer to another assessment offered, it is reassigned to the given assessment offered `Id`.

Parameters

- **`assessment_offered_id`** (`osid.id.Id`) – the `Id` of an `AssessmentOffered`
- **`alias_id`** (`osid.id.Id`) – the alias `Id`

Raise `AlreadyExists` – `alias_id` is in use as a primary `Id`

Raise `NotFound` – `assessment_offered_id` not found

Raise `NullArgument` – `assessment_offered_id` or `alias_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

Assessment Offered Bank Methods

`Bank.`**`can_lookup_assessment_offered_bank_mappings`**`()`

Tests if this user can perform lookups of assessment offered/bank mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known lookup methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if looking up mappings is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`use_comparative_bank_view`**`()`

The returns from the lookup methods may omit or translate elements based on this session, such as assessment, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`Bank.`**`use_plenary_bank_view`**`()`

A complete view of the `AssessmentTaken` and `Bank` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`Bank.`**`get_assessment_offered_ids_by_bank`**`(`*bank_id*`)`

Gets the list of `AssessmentOffered Ids` associated with a `Bank`.

Parameters **`bank_id`** (`osid.id.Id`) – `Id` of the `Bank`

Returns list of related assessment offered `Ids`

Return type `osid.id.IdList`

Raise `NotFound` – `bank_id` is not found

Raise `NullArgument` – `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_offered_by_bank`**`(`*bank_id*`)`

Gets the list of `AssessmentOffereds` associated with a `Bank`.

Parameters **`bank_id`** (`osid.id.Id`) – `Id` of the `Bank`

Returns list of related assessments offered

Return type `osid.assessment.AssessmentOfferedList`

Raise `NotFound` – `bank_id` is not found

Raise `NullArgument` – `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessment_offered_ids_by_banks`**(*bank_ids*)

Gets the list of `AssessmentOffered Ids` corresponding to a list of `Banks`.

Parameters **`bank_ids`** (`osid.id.IdList`) – list of bank `Ids`

Returns list of bank `Ids`

Return type `osid.id.IdList`

Raise `NullArgument` – `bank_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_offered_by_banks`**(*bank_ids*)

Gets the list of `AssessmentOffered` objects corresponding to a list of `Banks`.

Parameters **`bank_ids`** (`osid.id.IdList`) – list of bank `Ids`

Returns list of assessments offered

Return type `osid.assessment.AssessmentOfferedList`

Raise `NullArgument` – `bank_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_bank_ids_by_assessment_offered`**(*assessment_offered_id*)

Gets the list of `Bank Ids` mapped to an `AssessmentOffered`.

Parameters **`assessment_offered_id`** (`osid.id.Id`) – `Id` of an `AssessmentOffered`

Returns list of bank `Ids`

Return type `osid.id.IdList`

Raise `NotFound` – `assessment_offered_id` is not found

Raise `NullArgument` – `assessment_offered_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_banks_by_assessment_offered`**(*assessment_offered_id*)

Gets the list of `Banks` mapped to an `AssessmentOffered`.

Parameters **`assessment_offered_id`** (`osid.id.Id`) – `Id` of an `AssessmentOffered`

Returns list of banks

Return type `osid.assessment.BankList`

Raise `NotFound` – `assessment_offered_id` is not found

Raise `NullArgument` – `assessment_offered_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

Assessment Offered Bank Assignment Methods

`Bank.`**`can_assign_assessments_offered`**`()`

Tests if this user can alter assessment offered/bank mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known mapping methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if mapping is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`can_assign_assessments_offered_to_bank`**`(`*bank_id*`)`

Tests if this user can alter assessment offered/bank mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known mapping methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Parameters **`bank_id`** (`osid.id.Id`) – the `Id` of the `Bank`

Returns `false` if mapping is not authorized, `true` otherwise

Return type `boolean`

Raise `NullArgument` – `bank_id` is `null`

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assignable_bank_ids`**`(`*bank_id*`)`

Gets a list of banks including and under the given banks node in which any assessment taken can be assigned.

Parameters **`bank_id`** (`osid.id.Id`) – the `Id` of the `Bank`

Returns list of assignable bank `Ids`

Return type `osid.id.IdList`

Raise `NullArgument` – `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assignable_bank_ids_for_assessment_offered`**`(`*bank_id*, *assessment_offered_id*`)`

Gets a list of bank including and under the given bank node in which a specific assessment offered can be assigned.

Parameters

- **bank_id** (`osid.id.Id`) – the `Id` of the `Bank`
- **assessment_offered_id** (`osid.id.Id`) – the `Id` of the `AssessmentOffered`

Returns list of assignable bank `Ids`

Return type `osid.id.IdList`

Raise `NullArgument` – `bank_id` or `assessment_offered_id` is `null`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`Bank.`**`assign_assessment_offered_to_bank`**(*assessment_offered_id*, *bank_id*)

Adds an existing `AssessmentOffered` to a `Bank`.

Parameters

- **assessment_offered_id** (`osid.id.Id`) – the `Id` of the `AssessmentOffered`
- **bank_id** (`osid.id.Id`) – the `Id` of the `Bank`

Raise `AlreadyExists` – `assessment_offered_id` is already assigned to `bank_id`

Raise `NotFound` – `assessment_offered_id` or `bank_id` not found

Raise `NullArgument` – `assessment_offered_id` or `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`unassign_assessment_offered_from_bank`**(*assessment_offered_id*, *bank_id*)

Removes an `AssessmentOffered` from a `Bank`.

Parameters

- **assessment_offered_id** (`osid.id.Id`) – the `Id` of the `AssessmentOffered`
- **bank_id** (`osid.id.Id`) – the `Id` of the `Bank`

Raise `NotFound` – `assessment_offered_id` or `bank_id` not found or `assessment_offered_id` not assigned to `bank_id`

Raise `NullArgument` – `assessment_offered_id` or `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`reassign_assessment_offered_to_billing`**(*assessment_offered_id*, *from_bank_id*, *to_bank_id*)

Moves an `AssessmentOffered` from one `Bank` to another.

Mappings to other `Banks` are unaffected.

Parameters

- **assessment_offered_id** (`osid.id.Id`) – the `Id` of the `AssessmentOffered`
- **from_bank_id** (`osid.id.Id`) – the `Id` of the current `Bank`
- **to_bank_id** (`osid.id.Id`) – the `Id` of the destination `Bank`

Raise `NotFound` – `assessment_offered_id, from_bank_id,` or `to_bank_id` not found or `assessment_offered_id` not mapped to `from_bank_id`

Raise `NullArgument` – `assessment_offered_id, from_bank_id,` or `to_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Assessment Taken Lookup Methods

`Bank.`**`bank_id`**

Gets the `Bank Id` associated with this session.

Returns the `Bank Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Bank.`**`bank`**

Gets the `Bank` associated with this session.

Returns the `Bank` associated with this session

Return type `osid.assessment.Bank`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`can_lookup_assessments_taken`**`()`

Tests if this user can perform `AssessmentTaken` lookups.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if lookup methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`use_comparative_assessment_taken_view`**`()`

The returns from the lookup methods may omit or translate elements based on this session, such as assessment, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

Bank.**use_plenary_assessment_taken_view**()

A complete view of the `AssessmentTaken` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

Bank.**use_federated_bank_view**()

Federates the view for methods in this session.

A federated view will include assessments taken in banks which are children of this bank in the bank hierarchy.

compliance: mandatory – This method is must be implemented.

Bank.**use_isolated_bank_view**()

Isolates the view for methods in this session.

An isolated view restricts searches to this bank only.

compliance: mandatory – This method is must be implemented.

Bank.**get_assessment_taken**(*assessment_taken_id*)

Gets the `AssessmentTaken` specified by its `Id`.

In plenary mode, the exact `Id` is found or a `NotFound` results. Otherwise, the returned `AssessmentTaken` may have a different `Id` than requested, such as the case where a duplicate `Id` was assigned to an `AssessmentTaken` and retained for compatibility.

Parameters **assessment_taken_id** (`osid.id.Id`) – `Id` of the `AssessmentTaken`

Returns the assessment taken

Return type `osid.assessment.AssessmentTaken`

Raise `NotFound` – `assessment_taken_id` not found

Raise `NullArgument` – `assessment_taken_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method is must be implemented.

Bank.**get_assessments_taken_by_ids**(*assessment_taken_ids*)

Gets an `AssessmentTakenList` corresponding to the given `IdList`.

In plenary mode, the returned list contains all of the assessments specified in the `Id` list, in the order of the list, including duplicates, or an error results if an `Id` in the supplied list is not found or inaccessible. Otherwise, inaccessible `AssessmentTaken` objects may be omitted from the list and may present the elements in any order including returning a unique set.

Parameters **assessment_taken_ids** (`osid.id.IdList`) – the list of `Ids` to retrieve

Returns the returned `AssessmentTaken list`

Return type `osid.assessment.AssessmentTakenList`

Raise `NotFound` – an `Id was` not found

Raise `NullArgument` – `assessment_taken_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – assessment failure

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_taken_by_genus_type`**(*assessment_taken_genus_type*)
Gets an `AssessmentTakenList` corresponding to the given assessment taken genus `Type` which does not include assessments of types derived from the specified `Type`.

In plenary mode, the returned list contains all known assessments taken or an error results. Otherwise, the returned list may contain only those assessments taken that are accessible through this session.

Parameters **`assessment_taken_genus_type`** (`osid.type.Type`) – an assessment taken genus type

Returns the returned `AssessmentTaken list`

Return type `osid.assessment.AssessmentTakenList`

Raise `NullArgument` – `assessment_taken_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_taken_by_parent_genus_type`**(*assessment_taken_genus_type*)
Gets an `AssessmentTakenList` corresponding to the given assessment taken genus `Type` and include any additional assessments with genus types derived from the specified `Type`.

In plenary mode, the returned list contains all known assessments or an error results. Otherwise, the returned list may contain only those assessments taken that are accessible through this session.

Parameters **`assessment_taken_genus_type`** (`osid.type.Type`) – an assessment taken genus type

Returns the returned `AssessmentTaken list`

Return type `osid.assessment.AssessmentTakenList`

Raise `NullArgument` – `assessment_taken_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_taken_by_record_type`**(*assessment_taken_record_type*)
Gets an `AssessmentTakenList` corresponding to the given assessment taken record `Type`.

The set of assessments implementing the given record type is returned. In plenary mode, the returned list contains all known assessments taken or an error results. Otherwise, the returned list may contain only those assessments taken that are accessible through this session. In both cases, the order of the set is not specified.

Parameters **`assessment_taken_record_type`** (`osid.type.Type`) – an assessment taken record type

Returns the returned `AssessmentTaken` list

Return type `osid.assessment.AssessmentTakenList`

Raise `NullArgument` – `assessment_taken_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_taken_by_date`**(*from_*, *to*)

Gets an `AssessmentTakenList` started in the given date range inclusive.

In plenary mode, the returned list contains all known assessments taken or an error results. Otherwise, the returned list may contain only those assessments taken that are accessible through this session. In both cases, the order of the set is not specified.

Parameters

- **`from`** (`osid.calendaring.DateTime`) – start date
- **`to`** (`osid.calendaring.DateTime`) – end date

Returns the returned `AssessmentTaken` list

Return type `osid.assessment.AssessmentTakenList`

Raise `InvalidArgument` – `from` is greater than `to`

Raise `NullArgument` – `from` or `to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_taken_for_taker`**(*resource_id*)

Gets an `AssessmentTakenList` for the given resource.

In plenary mode, the returned list contains all known assessments taken or an error results. Otherwise, the returned list may contain only those assessments taken that are accessible through this session.

Parameters **`resource_id`** (`osid.id.Id`) – `Id` of a `Resource`

Returns the returned `AssessmentTaken` list

Return type `osid.assessment.AssessmentTakenList`

Raise `NullArgument` – `resource_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_taken_by_date_for_taker`**(*resource_id*, *from_*, *to*)

Gets an `AssessmentTakenList` started in the given date range inclusive for the given resource.

In plenary mode, the returned list contains all known assessments taken or an error results. Otherwise, the returned list may contain only those assessments taken that are accessible through this session.

Parameters

- **`resource_id`** (`osid.id.Id`) – `Id` of a `Resource`
- **`from`** (`osid.calendaring.DateTime`) – start date
- **`to`** (`osid.calendaring.DateTime`) – end date

Returns the returned `AssessmentTaken` list

Return type `osid.assessment.AssessmentTakenList`

Raise `InvalidArgument` – `from` is greater than `to`

Raise `NullArgument` – `resource_id, from` or `to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_taken_for_assessment`**(*assessment_id*)

Gets an `AssessmentTakenList` for the given assessment.

In plenary mode, the returned list contains all known assessments taken or an error results. Otherwise, the returned list may contain only those assessments taken that are accessible through this session.

Parameters **`assessment_id`** (`osid.id.Id`) – `Id` of an `Assessment`

Returns the returned `AssessmentTaken` list

Return type `osid.assessment.AssessmentTakenList`

Raise `NullArgument` – `assessment_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_taken_by_date_for_assessment`**(*assessment_id*, *from_*, *to*)

Gets an `AssessmentTakenList` started in the given date range inclusive for the given assessment.

In plenary mode, the returned list contains all known assessments taken or an error results. Otherwise, the returned list may contain only those assessments taken that are accessible through this session.

Parameters

- **`assessment_id`** (`osid.id.Id`) – `Id` of an `Assessment`
- **`from`** (`osid.calendaring.DateTime`) – start date
- **`to`** (`osid.calendaring.DateTime`) – end date

Returns the returned `AssessmentTaken` list

Return type `osid.assessment.AssessmentTakenList`

Raise `InvalidArgument` – `from` is greater than `to`

Raise `NullArgument` – `assessment_id, from` or `to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_taken_for_taker_and_assessment`**(*resource_id*, *assessment_id*)

Gets an `AssessmentTakenList` for the given resource and assessment.

In plenary mode, the returned list contains all known assessments taken or an error results. Otherwise, the returned list may contain only those assessments taken that are accessible through this session.

Parameters

- **resource_id** (osid.id.Id) – Id of a Resource
- **assessment_id** (osid.id.Id) – Id of an Assessment

Returns the returned AssessmentTaken list

Return type osid.assessment.AssessmentTakenList

Raise NullArgument – resource_id or assessment_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure occurred

compliance: mandatory – This method must be implemented.

Bank.**get_assessments_taken_by_date_for_taker_and_assessment**(*resource_id*, *assessment_id*, *from_*, *to*)

Gets an AssessmentTakenList started in the given date range inclusive for the given resource and assessment.

In plenary mode, the returned list contains all known assessments taken or an error results. Otherwise, the returned list may contain only those assessments taken that are accessible through this session.

Parameters

- **resource_id** (osid.id.Id) – Id of a Resource
- **assessment_id** (osid.id.Id) – Id of an Assessment
- **from** (osid.calendaring.DateTime) – start date
- **to** (osid.calendaring.DateTime) – end date

Returns the returned AssessmentTaken list

Return type osid.assessment.AssessmentTakenList

Raise InvalidArgument – from is greater than to

Raise NullArgument – resource_id, assessment_id, from or to is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure occurred

compliance: mandatory – This method must be implemented.

Bank.**get_assessments_taken_for_assessment_offered**(*assessment_offered_id*)

Gets an AssessmentTakenList by the given assessment offered.

In plenary mode, the returned list contains all known assessments taken or an error results. Otherwise, the returned list may contain only those assessments taken that are accessible through this session.

Parameters **assessment_offered_id** (osid.id.Id) – Id of an AssessmentOffered

Returns the returned `AssessmentTaken` list

Return type `osid.assessment.AssessmentTakenList`

Raise `NullArgument` – `assessment_offered_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_taken_by_date_for_assessment_offered`**(*assessment_offered_id*, *from_*, *to*)

Gets an `AssessmentTakenList` started in the given date range inclusive for the given assessment offered.

In plenary mode, the returned list contains all known assessments taken or an error results. Otherwise, the returned list may contain only those assessments taken that are accessible through this session.

Parameters

- **`assessment_offered_id`** (`osid.id.Id`) – `Id` of an `AssessmentOffered`
- **`from`** (`osid.calendaring.DateTime`) – start date
- **`to`** (`osid.calendaring.DateTime`) – end date

Returns the returned `AssessmentTaken` list

Return type `osid.assessment.AssessmentTakenList`

Raise `InvalidArgument` – `from` is greater than `to`

Raise `NullArgument` – `assessment_offered_id, from,` or `to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_taken_for_taker_and_assessment_offered`**(*resource_id*, *assessment_offered_id*)

Gets an `AssessmentTakenList` for the given resource and assessment offered.

In plenary mode, the returned list contains all known assessments taken or an error results. Otherwise, the returned list may contain only those assessments taken that are accessible through this session.

Parameters

- **`resource_id`** (`osid.id.Id`) – `Id` of a `Resource`
- **`assessment_offered_id`** (`osid.id.Id`) – `Id` of an `AssessmentOffered`

Returns the returned `AssessmentTaken` list

Return type `osid.assessment.AssessmentTakenList`

Raise `NullArgument` – `resource_id` or `assessmen_offeredt_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_taken_by_date_for_taker_and_assessment_offered`**(*resource_id, assessment_offered_id, from_, to*)

Gets an `AssessmentTakenList` started in the given date range inclusive for the given resource and assessment offered.

In plenary mode, the returned list contains all known assessments taken or an error results. Otherwise, the returned list may contain only those assessments taken that are accessible through this session.

Parameters

- **`resource_id`** (`osid.id.Id`) – `Id` of a `Resource`
- **`assessment_offered_id`** (`osid.id.Id`) – `Id` of an `AssessmentOffered`
- **`from`** (`osid.calendaring.DateTime`) – start date
- **`to`** (`osid.calendaring.DateTime`) – end date

Returns the returned `AssessmentTaken` list

Return type `osid.assessment.AssessmentTakenList`

Raise `InvalidArgument` – `from` is greater than `to`

Raise `NullArgument` – `resource_id, assessment_offered_id, from,` or `to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`assessments_taken`**

Gets all `AssessmentTaken` elements.

In plenary mode, the returned list contains all known assessments taken or an error results. Otherwise, the returned list may contain only those assessments taken that are accessible through this session.

Returns a list of `AssessmentTaken` elements

Return type `osid.assessment.AssessmentTakenList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

Assessment Taken Query Methods

`Bank.`**`bank_id`**

Gets the `Bank Id` associated with this session.

Returns the `Bank Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Bank.`**`bank`**

Gets the `Bank` associated with this session.

Returns the `Bank` associated with this session

Return type `osid.assessment.Bank`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`can_search_assessments_taken`**`()`

Tests if this user can perform `AssessmentTaken` searches.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer search operations to unauthorized users.

Returns `false` if search methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`use_federated_bank_view`**`()`

Federates the view for methods in this session.

A federated view will include assessments taken in banks which are children of this bank in the bank hierarchy.

compliance: mandatory – This method is must be implemented.

`Bank.`**`use_isolated_bank_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts searches to this bank only.

compliance: mandatory – This method is must be implemented.

`Bank.`**`assessment_taken_query`**

Gets an assessment taken query.

Returns the assessment taken query

Return type `osid.assessment.AssessmentTakenQuery`

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_taken_by_query`**`(`*assessment_taken_query*`)`

Gets a list of `AssessmentTaken` elements matching the given assessment taken query.

Parameters **`assessment_taken_query`** (`osid.assessment.AssessmentTakenQuery`) – the assessment taken query

Returns the returned `AssessmentTakenList`

Return type `osid.assessment.AssessmentTakenList`

Raise `NullArgument` – `assessment_taken_query` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – `assessment_taken_query` is not of this service

compliance: mandatory – This method must be implemented.

Assessment Taken Admin Methods

`Bank.`**`bank_id`**

Gets the `Bank Id` associated with this session.

Returns the `Bank Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Bank.`**`bank`**

Gets the `Bank` associated with this session.

Returns the `Bank` associated with this session

Return type `osid.assessment.Bank`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`can_create_assessments_taken`**`()`

Tests if this user can create `AssessmentTaken` objects.

A return of true does not guarantee successful authoriization. A return of false indicates that it is known creating an `AssessmentTaken` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer create operations to an unauthorized user.

Returns `false` if `AssessmentTaken` creation is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`can_create_assessment_taken_with_record_types`**`(`*assessment_taken_record_types*`)`

Tests if this user can create a single `AssessmentTaken` using the desired record types.

While `AssessmentManager.getAssessmentTakenRecordTypes()` can be used to examine which records are supported, this method tests which record(s) are required for creating a specific `AssessmentTaken`. Providing an empty array tests if an `AssessmentTaken` can be created with no records.

Parameters **`assessment_taken_record_types`** (`osid.type.Type[]`) – array of assessment taken record types

Returns `true` if `AssessmentTaken` creation using the specified record `Types` is supported, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `assessment_taken_record_types` is `null`

compliance: mandatory – This method must be implemented.

Bank.**get_assessment_taken_form_for_create**(*assessment_offered_id*, *assessment_taken_record_types*)

Gets the assessment taken form for creating new assessments taken.

A new form should be requested for each create transaction.

Parameters

- **assessment_offered_id** (osid.id.Id) – the Id of the related AssessmentOffered
- **assessment_taken_record_types** (osid.type.Type[]) – array of assessment taken record types to be included in the create operation or an empty list if none

Returns the assessment taken form

Return type osid.assessment.AssessmentTakenForm

Raise NotFound – assessment_offered_id is not found

Raise NullArgument – assessment_offered_id or assessment_taken_record_types is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure occurred

Raise Unsupported – unable to get form for requested record types

compliance: mandatory – This method must be implemented.

Bank.**create_assessment_taken**(*assessment_taken_form*)

Creates a new AssessmentTaken.

Parameters **assessment_taken_form** (osid.assessment.AssessmentTakenForm) – the form for this AssessmentTaken

Returns the new AssessmentTaken

Return type osid.assessment.AssessmentTaken

Raise IllegalState – assessment_taken_form already used in a create transaction

Raise InvalidArgument – one or more of the form elements is invalid

Raise NullArgument – assessment_taken_form is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure occurred

Raise Unsupported – assessment_offered_form did not originate from get_assessment_taken_form_for_create()

compliance: mandatory – This method must be implemented.

Bank.**can_update_assessments_taken**()

Tests if this user can update AssessmentTaken objects.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating an AssessmentTaken will result in a PermissionDenied. This is intended as a hint to an application that may opt not to offer update operations to an unauthorized user.

Returns false if AssessmentTaken modification is not authorized, true otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessment_taken_form_for_update`**`(`*assessment_taken_id*`)`

Gets the assessment taken form for updating an existing assessment taken.

A new assessment taken form should be requested for each update transaction.

Parameters **`assessment_taken_id`** (`osid.id.Id`) – the `Id` of the `AssessmentTaken`

Returns the assessment taken form

Return type `osid.assessment.AssessmentTakenForm`

Raise `NotFound` – `assessment_taken_id` is not found

Raise `NullArgument` – `assessment_taken_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`update_assessment_taken`**`(`*assessment_taken_form*`)`

Updates an existing assessment taken.

Parameters **`assessment_taken_form`** (`osid.assessment.AssessmentTakenForm`) – the form containing the elements to be updated

Raise `IllegalState` – `assessment_taken_form` already used in an update transaction

Raise `InvalidArgument` – the form contains an invalid value

Raise `NullArgument` – `assessment_taken_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – `assessment_offered_form` did not originate from `get_assessment_taken_form_for_update()`

compliance: mandatory – This method must be implemented.

`Bank.`**`can_delete_assessments_taken`**`()`

Tests if this user can delete `AssessmentsTaken`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting an `AssessmentTaken` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer a delete operations to unauthorized users.

Returns `false` if `AssessmentTaken` deletion is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`delete_assessment_taken`**`(`*assessment_taken_id*`)`

Deletes an `AssessmentTaken`.

Parameters **`assessment_taken_id`** (`osid.id.Id`) – the `Id` of the `AssessmentTaken` to remove

Raise `NotFound` – `assessment_taken_id` not found

Raise `NullArgument` – `assessment_taken_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`can_manage_assessment_taken_aliases`**`()`

Tests if this user can manage `Id` aliases for `AssessmentsTaken`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known changing an alias will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer alias operations to an unauthorized user.

Returns `false` if `AssessmentTaken` aliasing is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`alias_assessment_taken`**`(`*assessment_taken_id*, *alias_id*`)`

Adds an `Id` to an `AssessmentTaken` for the purpose of creating compatibility.

The primary `Id` of the `AssessmentTaken` is determined by the provider. The new `Id` is an alias to the primary `Id`. If the alias is a pointer to another assessment taken, it is reassigned to the given assessment taken `Id`.

Parameters

- **`assessment_taken_id`** (`osid.id.Id`) – the `Id` of an `AssessmentTaken`
- **`alias_id`** (`osid.id.Id`) – the alias `Id`

Raise `AlreadyExists` – `alias_id` is in use as a primary `Id`

Raise `NotFound` – `assessment_taken_id` not found

Raise `NullArgument` – `assessment_taken_id` or `alias_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

Assessment Taken Bank Methods

`Bank.`**`can_lookup_assessment_taken_bank_mappings`**`()`

Tests if this user can perform lookups of assessment taken/bank mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known lookup methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if looking up mappings is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`use_comparative_bank_view`**`()`

The returns from the lookup methods may omit or translate elements based on this session, such as assessment, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`Bank.`**`use_plenary_bank_view`**`()`

A complete view of the `AssessmentTaken` and `Bank` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`Bank.`**`get_assessment_taken_ids_by_bank`**`(`*bank_id*`)`

Gets the list of `AssessmentTaken Ids` associated with a `Bank`.

Parameters **`bank_id`** (`osid.id.Id`) – `Id` of the `Bank`

Returns list of related assessment taken `Ids`

Return type `osid.id.IdList`

Raise `NotFound` – `bank_id` is not found

Raise `NullArgument` – `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_taken_by_bank`**`(`*bank_id*`)`

Gets the list of `AssessmentTakens` associated with a `Bank`.

Parameters **`bank_id`** (`osid.id.Id`) – `Id` of the `Bank`

Returns list of related assessments taken

Return type `osid.assessment.AssessmentTakenList`

Raise `NotFound` – `bank_id` is not found

Raise `NullArgument` – `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessment_taken_ids_by_banks`**`(`*bank_ids*`)`

Gets the list of `AssessmentTaken Ids` corresponding to a list of `Banks`.

Parameters **`bank_ids`** (`osid.id.IdList`) – list of bank `Ids`

Returns list of bank `Ids`

Return type `osid.id.IdList`

Raise `NullArgument` – `bank_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assessments_taken_by_banks`**`(`*bank_ids*`)`

Gets the list of `AssessmentTaken` objects corresponding to a list of `Banks`.

Parameters **`bank_ids`** (`osid.id.IdList`) – list of bank `Ids`

Returns list of assessments taken

Return type `osid.assessment.AssessmentTakenList`

Raise `NullArgument` – `bank_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_bank_ids_by_assessment_taken`**`(`*assessment_taken_id*`)`

Gets the list of `Bank Ids` mapped to an `AssessmentTaken`.

Parameters **`assessment_taken_id`** (`osid.id.Id`) – `Id` of an `AssessmentTaken`

Returns list of bank `Ids`

Return type `osid.id.IdList`

Raise `NotFound` – `assessment_taken_id` is not found

Raise `NullArgument` – `assessment_taken_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`get_banks_by_assessment_taken`**`(`*assessment_taken_id*`)`

Gets the list of `Banks` mapped to an `AssessmentTaken`.

Parameters **`assessment_taken_id`** (`osid.id.Id`) – `Id` of an `AssessmentTaken`

Returns list of banks

Return type `osid.assessment.BankList`

Raise `NotFound` – `assessment_taken_id` is not found

Raise `NullArgument` – `assessment_taken_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

Assessment Taken Bank Assignment Methods

`Bank.`**`can_assign_assessments_taken`**`()`

Tests if this user can alter assessment taken/bank mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known mapping methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if mapping is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bank.`**`can_assign_assessments_taken_to_bank`**(*bank_id*)

Tests if this user can alter assessment taken/bank mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known mapping methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Parameters **`bank_id`** (`osid.id.Id`) – the `Id` of the `Bank`

Returns `false` if mapping is not authorized, `true` otherwise

Return type `boolean`

Raise `NullArgument` – `bank_id` is `null`

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assignable_bank_ids`**(*bank_id*)

Gets a list of banks including and under the given banks node in which any assessment taken can be assigned.

Parameters **`bank_id`** (`osid.id.Id`) – the `Id` of the `Bank`

Returns list of assignable bank `Ids`

Return type `osid.id.IdList`

Raise `NullArgument` – `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`Bank.`**`get_assignable_bank_ids_for_assessment_taken`**(*bank_id*, *assessment_taken_id*)

Gets a list of bank including and under the given bank node in which a specific assessment taken can be assigned.

Parameters

- **`bank_id`** (`osid.id.Id`) – the `Id` of the `Bank`
- **`assessment_taken_id`** (`osid.id.Id`) – the `Id` of the `AssessmentTaken`

Returns list of assignable bank `Ids`

Return type `osid.id.IdList`

Raise `NullArgument` – `bank_id` or `assessment_taken_id` is `null`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`Bank.`**`assign_assessment_taken_to_bank`**(*assessment_taken_id*, *bank_id*)

Adds an existing `AssessmentTaken` to a `Bank`.

Parameters

- **`assessment_taken_id`** (`osid.id.Id`) – the `Id` of the `AssessmentTaken`
- **`bank_id`** (`osid.id.Id`) – the `Id` of the `Bank`

Raise `AlreadyExists` – `assessment_taken_id` is already assigned to `bank_id`

Raise `NotFound` – `assessment_taken_id` or `bank_id` not found

Raise `NullArgument` – `assessment_taken_id` or `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`unassign_assessment_taken_from_bank`**(*assessment_taken_id*, *bank_id*)

Removes an `AssessmentTaken` from a `Bank`.

Parameters

- **`assessment_taken_id`** (`osid.id.Id`) – the `Id` of the `AssessmentTaken`
- **`bank_id`** (`osid.id.Id`) – the `Id` of the `Bank`

Raise `NotFound` – `assessment_taken_id` or `bank_id` not found or `assessment_taken_id` not assigned to `bank_id`

Raise `NullArgument` – `assessment_taken_id` or `bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

compliance: mandatory – This method must be implemented.

`Bank.`**`reassign_assessment_taken_to_billing`**(*assessment_taken_id*, *from_bank_id*, *to_bank_id*)

Moves an `AssessmentTaken` from one `Bank` to another.

Mappings to other `Banks` are unaffected.

Parameters

- **`assessment_taken_id`** (`osid.id.Id`) – the `Id` of the `AssessmentTaken`
- **`from_bank_id`** (`osid.id.Id`) – the `Id` of the current `Bank`
- **`to_bank_id`** (`osid.id.Id`) – the `Id` of the destination `Bank`

Raise `NotFound` – `assessment_taken_id, from_bank_id,` or `to_bank_id` not found or `assessment_taken_id` not mapped to `from_bank_id`

Raise `NullArgument` – `assessment_taken_id, from_bank_id,` or `to_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Objects

Question

class `dlkit.assessment.objects.`**`Question`**

Bases: *`dlkit.osid.objects.OsidObject`*

A `Question` represents the question portion of an assessment item.

Like all OSID objects, a `Question` is identified by its `Id` and any persisted references should use the `Id`.

`get_question_record`(*question_record_type*)
Gets the item record corresponding to the given `Question` record `Type`.

This method is used to retrieve an object implementing the requested record. The `question_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(question_record_type)` is `true` .

Parameters **`question_record_type`** (`osid.type.Type`) – the type of the record to retrieve

Returns the question record

Return type `osid.assessment.records.QuestionRecord`

Raise `NullArgument` – `question_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(question_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Question Form

class `dlkit.assessment.objects.`**`QuestionForm`**
Bases: *`dlkit.osid.objects.OsidObjectForm`*

This is the form for creating and updating `Questions`.

`get_question_form_record`(*question_record_type*)
Gets the `QuestionFormRecord` corresponding to the given question record `Type`.

Parameters **`question_record_type`** (`osid.type.Type`) – the question record type

Returns the question record

Return type `osid.assessment.records.QuestionFormRecord`

Raise `NullArgument` – `question_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(question_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Question List

class `dlkit.assessment.objects.`**`QuestionList`**
Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists, QuestionList` provides a means for accessing `Question` elements sequentially either one at a time or many at a time.

Examples: while (ql.hasNext()) { Question question = ql.getNextQuestion(); }

or

while (ql.hasNext()) { Question[] question = al.getNextQuestions(ql.available());

}

next_question

Gets the next `Question` in this list.

Returns the next `Question` in this list. The `has_next()` method should be used to test that a next `Question` is available before calling this method.

Return type `osid.assessment.Question`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

get_next_questions(*n*)

Gets the next set of `Question` elements in this list which must be less than or equal to the number returned from `available()`.

Parameters n (`cardinal`) – the number of `Question` elements requested which should be less than or equal to `available()`

Returns an array of `Question` elements.The length of the array is less than or equal to the number specified.

Return type `osid.assessment.Question`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Answer

class `dlkit.assessment.objects.`**Answer**

Bases: *dlkit.osid.objects.OsidObject*

An `Answer` represents the question portion of an assessment item.

Like all OSID objects, an `Answer` is identified by its `Id` and any persisted references should use the `Id`.

get_answer_record(*answer_record_type*)

Gets the answer record corresponding to the given `Answer` record `Type`.

This method is used to retrieve an object implementing the requested records. The `answer_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(answer_record_type)` is `true` .

Parameters answer_record_type (`osid.type.Type`) – the type of the record to retrieve

Returns the answer record

Return type `osid.assessment.records.AnswerRecord`

Raise `NullArgument` – `answer_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(answer_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Answer Form

class `dlkit.assessment.objects.`**`AnswerForm`**

Bases: *`dlkit.osid.objects.OsidObjectForm`*

This is the form for creating and updating `Answers`.

`get_answer_form_record`(*answer_record_type*)

Gets the `AnswerFormRecord` corresponding to the given answer record `Type`.

Parameters **`answer_record_type`** (`osid.type.Type`) – the answer record type

Returns the answer record

Return type `osid.assessment.records.AnswerFormRecord`

Raise `NullArgument` – `answer_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(answer_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Answer List

class `dlkit.assessment.objects.`**`AnswerList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists`, `AnswerList` provides a means for accessing `Answer` elements sequentially either one at a time or many at a time.

Examples: while (al.hasNext()) { Answer answer = al.getNextAnswer(); }

or

while (al.hasNext()) { Answer[] answer = al.getNextAnswers(al.available());

}

`next_answer`

Gets the next `Answer` in this list.

Returns the next `Answer` in this list. The `has_next()` method should be used to test that a next `Answer` is available before calling this method.

Return type `osid.assessment.Answer`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_answers`(*n*)

Gets the next set of `Answer` elements in this list which must be less than or equal to the number returned from `available()`.

Parameters **`n`** (`cardinal`) – the number of `Answer` elements requested which should be less than or equal to `available()`

Returns an array of `Answer` elements.The length of the array is less than or equal to the number specified.

Return type `osid.assessment.Answer`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Item

class `dlkit.assessment.objects.`**`Item`**

Bases: *`dlkit.osid.objects.OsidObject`*, *`dlkit.osid.markers.Aggregateable`*

An `Item` represents an individual assessment item such as a question.

Like all OSID objects, a `Item` is identified by its `Id` and any persisted references should use the `Id`.

An `Item` is composed of a `Question` and an `Answer`.

`learning_objective_ids`

Gets the `Ids` of any `Objectives` corresponding to this item.

Returns the learning objective `Ids`

Return type `osid.id.IdList`

compliance: mandatory – This method must be implemented.

`learning_objectives`

Gets the any `Objectives` corresponding to this item.

Returns the learning objectives

Return type `osid.learning.ObjectiveList`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`question_id`

Gets the `Id` of the `Question`.

Returns the question `Id`

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`question`

Gets the question.

Returns the question

Return type `osid.assessment.Question`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`answer_ids`

Gets the `Ids` of the answers.

Questions may have more than one acceptable answer.

Returns the answer `Ids`

Return type `osid.id.IdList`

compliance: mandatory – This method must be implemented.

answers

Gets the answers.

Returns the answers

Return type `osid.assessment.AnswerList`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

get_item_record(*item_record_type*)

Gets the item record corresponding to the given `Item` record `Type`.

This method is used to retrieve an object implementing the requested records. The `item_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(item_record_type)` is `true` .

Parameters **item_record_type** (`osid.type.Type`) – the type of the record to retrieve

Returns the item record

Return type `osid.assessment.records.ItemRecord`

Raise `NullArgument` – `item_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(item_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Item Form

class `dlkit.assessment.objects.`**ItemForm**

Bases: *dlkit.osid.objects.OsidObjectForm*, *dlkit.osid.objects.OsidAggregateableForm*

This is the form for creating and updating `Items`.

Like all `OsidForm` objects, various data elements may be set here for use in the create and update methods in the `ItemAdminSession`. For each data element that may be set, metadata may be examined to provide display hints or data constraints.

learning_objectives_metadata

Gets the metadata for learning objectives.

Returns metadata for the learning objectives

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

learning_objectives

get_item_form_record(*item_record_type*)

Gets the `ItemnFormRecord` corresponding to the given item record `Type`.

Parameters **item_record_type** (`osid.type.Type`) – the item record type

Returns the item record

Return type `osid.assessment.records.ItemFormRecord`

Raise `NullArgument` – `item_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(item_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Item List

class `dlkit.assessment.objects.`**`ItemList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists,` `ItemList` provides a means for accessing `Item` elements sequentially either one at a time or many at a time.

Examples: while (il.hasNext()) { Item item = il.getNextItem(); }

or

while (il.hasNext()) { Item[] items = il.getNextItems(il.available());

}

`next_item`

Gets the next `Item` in this list.

Returns the next `Item` in this list. The `has_next()` method should be used to test that a next `Item` is available before calling this method.

Return type `osid.assessment.Item`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_items`(*n*)

Gets the next set of `Item` elements in this list which must be less than or equal to the number returned from `available()`.

Parameters **n** (`cardinal`) – the number of `Item` elements requested which should be less than or equal to `available()`

Returns an array of `Item` elements.The length of the array is less than or equal to the number specified.

Return type `osid.assessment.Item`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Assessment

class `dlkit.assessment.objects.`**`Assessment`**

Bases: *`dlkit.osid.objects.OsidObject`*

An `Assessment` represents a sequence of assessment items.

Like all OSID objects, an `Assessment` is identified by its `Id` and any persisted references should use the `Id`.

An `Assessment` may have an accompanying rubric used for assessing performance. The rubric assessment is established canonically in this `Assessment`.

`level_id`

Gets the `Id` of a `Grade` corresponding to the assessment difficulty.

Returns a grade `Id`

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`level`

Gets the `Grade` corresponding to the assessment difficulty.

Returns the level

Return type `osid.grading.Grade`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`has_rubric``()`

Tests if a rubric assessment is associated with this assessment.

Returns `true` if a rubric is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`rubric_id`

Gets the `Id` of the rubric.

Returns an assessment `Id`

Return type `osid.id.Id`

Raise `IllegalState` – `has_rubric()` is `false`

compliance: mandatory – This method must be implemented.

`rubric`

Gets the rubric.

Returns the assessment

Return type `osid.assessment.Assessment`

Raise `IllegalState` – `has_rubric()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_assessment_record``(`*assessment_record_type*`)`

Gets the assessment record corresponding to the given `Assessment` record `Type`.

This method is used to retrieve an object implementing the requested record. The `assessment_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(assessment_record_type)` is `true` .

Parameters **`assessment_record_type`** (`osid.type.Type`) – the type of the record to retrieve

Returns the assessment record

Return type `osid.assessment.records.AssessmentRecord`

Raise `NullArgument` – `assessment_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(assessment_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Assessment Form

class `dlkit.assessment.objects.`**`AssessmentForm`**

Bases: *`dlkit.osid.objects.OsidObjectForm`*

This is the form for creating and updating `Assessments`.

Like all `OsidForm` objects, various data elements may be set here for use in the create and update methods in the `AssessmentAdminSession`. For each data element that may be set, metadata may be examined to provide display hints or data constraints.

`level_metadata`

Gets the metadata for a grade level.

Returns metadata for the grade level

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`level`

`rubric_metadata`

Gets the metadata for a rubric assessment.

Returns metadata for the assesment

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`rubric`

`get_assessment_form_record`(*assessment_record_type*)

Gets the `AssessmentFormRecord` corresponding to the given assessment record `Type`.

Parameters **`assessment_record_type`** (`osid.type.Type`) – the assessment record type

Returns the assessment record

Return type `osid.assessment.records.AssessmentFormRecord`

Raise `NullArgument` – `assessment_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(assessment_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Assessment List

class `dlkit.assessment.objects.`**`AssessmentList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists, AssessmentList` provides a means for accessing `Assessment` elements sequentially either one at a time or many at a time.

Examples: while (al.hasNext()) { Assessment assessment = al.getNextAssessment(); }

or

while (al.hasNext()) { Assessment[] assessments = al.hetNextAssessments(al.available());

}

`next_assessment`

Gets the next `Assessment` in this list.

Returns the next `Assessment` in this list. The `has_next()` method should be used to test that a next `Assessment` is available before calling this method.

Return type `osid.assessment.Assessment`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_assessments``(`*n*`)`

Gets the next set of `Assessment` elements in this list which must be less than or equal to the number returned from `available()`.

Parameters **`n`** (`cardinal`) – the number of `Assessment` elements requested which should be less than or equal to `available()`

Returns an array of `Assessment` elements.The length of the array is less than or equal to the number specified.

Return type `osid.assessment.Assessment`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Assessment Offered

class `dlkit.assessment.objects.`**`AssessmentOffered`**

Bases: *`dlkit.osid.objects.OsidObject`*, *`dlkit.osid.markers.Subjugateable`*

An `AssessmentOffered` represents a sequence of assessment items.

Like all OSID objects, an `AssessmentOffered` is identified by its `Id` and any persisted references should use the `Id`.

`assessment_id`

Gets the assessment `Id` corresponding to this assessment offering.

Returns the assessment id

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`assessment`
Gets the assessment corresponding to this assessment offereng.

Returns the assessment

Return type `osid.assessment.Assessment`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`level_id`
Gets the `Id` of a `Grade` corresponding to the assessment difficulty.

Returns a grade id

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`level`
Gets the `Grade` corresponding to the assessment difficulty.

Returns the level

Return type `osid.grading.Grade`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`are_items_sequential``()`
Tests if the items or parts in this assessment are taken sequentially.

Returns `true` if the items are taken sequentially, `false` if the items can be skipped and revisited

Return type `boolean`

compliance: mandatory – This method must be implemented.

`are_items_shuffled``()`
Tests if the items or parts appear in a random order.

Returns `true` if the items appear in a random order, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`has_start_time``()`
Tests if there is a fixed start time for this assessment.

Returns `true` if there is a fixed start time, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`start_time`
Gets the start time for this assessment.

Returns the designated start time

Return type `osid.calendaring.DateTime`

Raise `IllegalState` – `has_start_time()` is `false`

compliance: mandatory – This method must be implemented.

has_deadline()
Tests if there is a fixed end time for this assessment.

Returns `true` if there is a fixed end time, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

deadline
Gets the end time for this assessment.

Returns the designated end time

Return type `osid.calendaring.DateTime`

Raise `IllegalState` – `has_deadline()` is `false`

compliance: mandatory – This method must be implemented.

has_duration()
Tests if there is a fixed duration for this assessment.

Returns `true` if there is a fixed duration, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

duration
Gets the duration for this assessment.

Returns the duration

Return type `osid.calendaring.Duration`

Raise `IllegalState` – `has_duration()` is `false`

compliance: mandatory – This method must be implemented.

is_scored()
Tests if this assessment will be scored.

Returns `true` if this assessment will be scored `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

score_system_id
Gets the grade system `Id` for the score.

Returns the grade system `Id`

Return type `osid.id.Id`

Raise `IllegalState` – `is_scored()` is `false`

compliance: mandatory – This method must be implemented.

score_system
Gets the grade system for the score.

Returns the grade system

Return type `osid.grading.GradeSystem`

Raise `IllegalState` – `is_scored()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

is_graded()

Tests if this assessment will be graded.

Returns `true` if this assessment will be graded, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

grade_system_id

Gets the grade system `Id` for the grade.

Returns the grade system `Id`

Return type `osid.id.Id`

Raise `IllegalState` – `is_graded()` is `false`

compliance: mandatory – This method must be implemented.

grade_system

Gets the grade system for the grade.

Returns the grade system

Return type `osid.grading.GradeSystem`

Raise `IllegalState` – `is_graded()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

has_rubric()

Tests if a rubric assessment is associated with this assessment.

Returns `true` if a rubric is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

rubric_id

Gets the `Id` of the rubric.

Returns an assessment offered `Id`

Return type `osid.id.Id`

Raise `IllegalState` – `has_rubric()` is `false`

compliance: mandatory – This method must be implemented.

rubric

Gets the rubric.

Returns the assessment offered

Return type `osid.assessment.AssessmentOffered`

Raise `IllegalState` – `has_rubric()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

get_assessment_offered_record(*assessment_taken_record_type*)

Gets the assessment offered record corresponding to the given `AssessmentOffered` record `Type`.

This method is used to retrieve an object implementing the requested record. The `assessment_offered_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(assessment_offered_record_type)` is `true` .

Parameters **assessment_taken_record_type** (`osid.type.Type`) – an assessment offered record type

Returns the assessment offered record

Return type `osid.assessment.records.AssessmentOfferedRecord`

Raise `NullArgument` – `assessment_offered_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(assessment_offered_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Assessment Offered Form

class `dlkit.assessment.objects.`**AssessmentOfferedForm**

Bases: *dlkit.osid.objects.OsidObjectForm*, *dlkit.osid.objects.OsidSubjugateableForm*

This is the form for creating and updating an `AssessmentOffered`.

Like all `OsidForm` objects, various data elements may be set here for use in the create and update methods in the `AssessmentOfferedAdminSession`. For each data element that may be set, metadata may be examined to provide display hints or data constraints.

level_metadata

Gets the metadata for a grade level.

Returns metadata for the grade level

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

level

items_sequential_metadata

Gets the metadata for sequential operation.

Returns metadata for the sequential flag

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

items_sequential

items_shuffled_metadata

Gets the metadata for shuffling items.

Returns metadata for the shuffled flag

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`items_shuffled`

`start_time_metadata`

Gets the metadata for the assessment start time.

Returns metadata for the start time

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`start_time`

`deadline_metadata`

Gets the metadata for the assessment deadline.

Returns metadata for the end time

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`deadline`

`duration_metadata`

Gets the metadata for the assessment duration.

Returns metadata for the duration

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`duration`

`score_system_metadata`

Gets the metadata for a score system.

Returns metadata for the grade system

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`score_system`

`grade_system_metadata`

Gets the metadata for a grading system.

Returns metadata for the grade system

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`grade_system`

`get_assessment_offered_form_record`(*assessment_offered_record_type*)

Gets the `AssessmentOfferedFormRecord` corresponding to the given assessment record `Type`.

Parameters **`assessment_offered_record_type`** (`osid.type.Type`) – the assessment offered record type

Returns the assessment offered record

Return type `osid.assessment.records.AssessmentOfferedFormRecord`

Raise `NullArgument` – `assessment_offered_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(assessment_offered_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Assessment Offered List

class `dlkit.assessment.objects.`**`AssessmentOfferedList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists, AssessmentOfferedList` provides a means for accessing `AssessmentTaken` elements sequentially either one at a time or many at a time.

Examples: while (aol.hasNext()) { AssessmentOffered assessment = aol.getNextAssessmentOffered();

or

while (aol.hasNext()) { AssessmentOffered[] assessments = aol.hetNextAssessmentsOffered(aol.available());

}

`next_assessment_offered`

Gets the next `AssessmentOffered` in this list.

Returns the next `AssessmentOffered` in this list. The `has_next()` method should be used to test that a next `AssessmentOffered` is available before calling this method.

Return type `osid.assessment.AssessmentOffered`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_assessments_offered`(*n*)

Gets the next set of `AssessmentOffered` elements in this list which must be less than or equal to the number returned from `available()`.

Parameters **n** (`cardinal`) – the number of `AssessmentOffered` elements requested which should be less than or equal to `available()`

Returns an array of `AssessmentOffered` elements.The length of the array is less than or equal to the number specified.

Return type `osid.assessment.AssessmentOffered`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Assessment Taken

class `dlkit.assessment.objects.`**`AssessmentTaken`**

Bases: *`dlkit.osid.objects.OsidObject`*

Represents a taken assessment or an assessment in progress.

assessment_offered_id

Gets the `Id` of the `AssessmentOffered`.

Returns the assessment offered `Id`

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

assessment_offered

Gets the `AssessmentOffered`.

Returns the assessment offered

Return type `osid.assessment.AssessmentOffered`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

taker_id

Gets the `Id` of the resource who took or is taking this assessment.

Returns the resource `Id`

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

taker

Gets the `Resource` taking this assessment.

Returns the resource

Return type `osid.resource.Resource`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

taking_agent_id

Gets the `Id` of the `Agent` who took or is taking the assessment.

Returns the agent `Id`

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

taking_agent

Gets the `Agent`.

Returns the agent

Return type `osid.authentication.Agent`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

has_started()

Tests if this assessment has begun.

Returns `true` if the assessment has begun, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

actual_start_time

Gets the time this assessment was started.

Returns the start time

Return type `osid.calendaring.DateTime`

Raise `IllegalState` – `has_started()` is `false`

compliance: mandatory – This method must be implemented.

has_ended()

Tests if this assessment has ended.

Returns `true` if the assessment has ended, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

completion_time

Gets the time of this assessment was completed.

Returns the end time

Return type `osid.calendaring.DateTime`

Raise `IllegalState` – `has_ended()` is `false`

compliance: mandatory – This method must be implemented.

time_spent

Gets the total time spent taking this assessment.

Returns the total time spent

Return type `osid.calendaring.Duration`

compliance: mandatory – This method must be implemented.

completion

Gets a completion percentage of the assessment.

Returns the percent complete (0-100)

Return type `cardinal`

compliance: mandatory – This method must be implemented.

is_scored()

Tests if a score is available for this assessment.

Returns `true` if a score is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

score_system_id

Gets a score system `Id` for the assessment.

Returns the grade system

Return type `osid.id.Id`

Raise `IllegalState` – `is_score()` is `false`

compliance: mandatory – This method must be implemented.

`score_system`

Gets a grade system for the score.

Returns the grade system

Return type `osid.grading.GradeSystem`

Raise `IllegalState` – `is_scored()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`score`

Gets a score for the assessment.

Returns the score

Return type `decimal`

Raise `IllegalState` – `is_scored()` is `false`

compliance: mandatory – This method must be implemented.

`is_graded``()`

Tests if a grade is available for this assessment.

Returns `true` if a grade is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`grade_id`

Gets a grade `Id` for the assessment.

Returns the grade

Return type `osid.id.Id`

Raise `IllegalState` – `is_graded()` is `false`

compliance: mandatory – This method must be implemented.

`grade`

Gets a grade for the assessment.

Returns the grade

Return type `osid.grading.Grade`

Raise `IllegalState` – `is_graded()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`feedback`

Gets any overall comments available for this assessment by the grader.

Returns comments

Return type `osid.locale.DisplayText`

compliance: mandatory – This method must be implemented.

has_rubric()

Tests if a rubric assessment is associated with this assessment.

Returns true if a rubric is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

rubric_id

Gets the Id of the rubric.

Returns an assessment taken Id

Return type osid.id.Id

Raise IllegalState – has_rubric() is false

compliance: mandatory – This method must be implemented.

rubric

Gets the rubric.

Returns the assessment taken

Return type osid.assessment.AssessmentTaken

Raise IllegalState – has_rubric() is false

Raise OperationFailed – unable to complete request

compliance: mandatory – This method must be implemented.

get_assessment_taken_record(*assessment_taken_record_type*)

Gets the assessment taken record corresponding to the given AssessmentTaken record Type.

This method is used to retrieve an object implementing the requested record. The assessment_taken_record_type may be the Type returned in get_record_types() or any of its parents in a Type hierarchy where has_record_type(assessment_taken_record_type) is true .

Parameters **assessment_taken_record_type** (osid.type.Type) – an assessment taken record type

Returns the assessment taken record

Return type osid.assessment.records.AssessmentTakenRecord

Raise NullArgument – assessment_taken_record_type is null

Raise OperationFailed – unable to complete request

Raise Unsupported – has_record_type(assessment_taken_record_type) is false

compliance: mandatory – This method must be implemented.

Assessment Taken Form

class dlkit.assessment.objects.**AssessmentTakenForm**

Bases: *dlkit.osid.objects.OsidObjectForm*

This is the form for creating and updating an AssessmentTaken.

Like all `OsidForm` objects, various data elements may be set here for use in the create and update methods in the `AssessmentTakenAdminSession`. For each data element that may be set, metadata may be examined to provide display hints or data constraints.

`taker_metadata`
Gets the metadata for a resource to manually set which resource will be taking the assessment.

Returns metadata for the resource

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`taker`

`get_assessment_taken_form_record`(*assessment_taken_record_type*)
Gets the `AssessmentTakenFormRecord` corresponding to the given assessment taken record `Type`.

Parameters **`assessment_taken_record_type`** (`osid.type.Type`) – the assessment taken record type

Returns the assessment taken record

Return type `osid.assessment.records.AssessmentTakenFormRecord`

Raise `NullArgument` – `assessment_taken_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(assessment_taken_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Assessment Taken List

class `dlkit.assessment.objects.`**`AssessmentTakenList`**
Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists, AssessmentTakenList` provides a means for accessing `AssessmentTaken` elements sequentially either one at a time or many at a time.

Examples: while (atl.hasNext()) { AssessmentTaken assessment = atl.getNextAssessmentTaken();

or

while (atl.hasNext()) { AssessmentTaken[] assessments = atl.hetNextAssessmentsTaken(atl.available());

}

`next_assessment_taken`
Gets the next `AssessmentTaken` in this list.

Returns the next `AssessmentTaken` in this list. The `has_next()` method should be used to test that a next `AssessmentTaken` is available before calling this method.

Return type `osid.assessment.AssessmentTaken`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

get_next_assessments_taken(*n*)

Gets the next set of `AssessmentTaken` elements in this list which must be less than or equal to the number returned from `available()`.

Parameters **n** (`cardinal`) – the number of `AssessmentTaken` elements requested which should be less than or equal to `available()`

Returns an array of `AssessmentTaken` elements.The length of the array is less than or equal to the number specified.

Return type `osid.assessment.AssessmentTaken`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Assessment Section

class `dlkit.assessment.objects.`**AssessmentSection**

Bases: *`dlkit.osid.objects.OsidObject`*

Represents an assessment section.

An assessment section represents a cluster of questions used to organize the execution of an assessment. The section is the student aspect of an assessment part.

assessment_taken_id

Gets the `Id` of the `AssessmentTaken`.

Returns the assessment taken `Id`

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

assessment_taken

Gets the `AssessmentTakeb`.

Returns the assessment taken

Return type `osid.assessment.AssessmentTaken`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

has_allocated_time()

Tests if this section must be completed within an allocated time.

Returns `true` if this section has an allocated time, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

allocated_time

Gets the allocated time for this section.

Returns allocated time

Return type `osid.calendaring.Duration`

Raise `IllegalState` – `has_allocated_time()` is `false`

compliance: mandatory – This method must be implemented.

are_items_sequential()

Tests if the items or parts in this section are taken sequentially.

Returns `true` if the items are taken sequentially, `false` if the items can be skipped and revisited

Return type `boolean`

compliance: mandatory – This method must be implemented.

are_items_shuffled()

Tests if the items or parts appear in a random order.

Returns `true` if the items appear in a random order, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

get_assessment_section_record(*assessment_section_record_type*)

Gets the assessment section record corresponding to the given `AssessmentSection` record `Type`.

This method is used to retrieve an object implementing the requested record. The `assessment_section_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(assessment_section_record_type)` is `true` .

Parameters **assessment_section_record_type** (`osid.type.Type`) – an assessment section record type

Returns the assessment section record

Return type `osid.assessment.records.AssessmentSectionRecord`

Raise `NullArgument` – `assessment_section_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(assessment_section_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Assessment Section List

class dlkit.assessment.objects.**AssessmentSectionList**

Bases: *dlkit.osid.objects.OsidList*

Like all `OsidLists`, `AssessmentSectionList` provides a means for accessing `AssessmentSection` elements sequentially either one at a time or many at a time.

Examples: while (asl.hasNext()) { AssessmentSection section = asl.getNextAssessmentSection();

or

while (asl.hasNext()) { AssessmentSection[] sections = asl.hetNextAssessmentSections(asl.available());

}

next_assessment_section

Gets the next `AssessmentSection` in this list.

Returns the next `AssessmentSection` in this list. The `has_next()` method should be used to test that a next `AssessmentSection` is available before calling this method.

Return type `osid.assessment.AssessmentSection`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_assessment_sections`(*n*)

Gets the next set of `AssessmentSection` elements in this list which must be less than or equal to the number returned from `available()`.

Parameters **`n`** (`cardinal`) – the number of `AssessmentSection` elements requested which should be less than or equal to `available()`

Returns an array of `AssessmentSection` elements.The length of the array is less than or equal to the number specified.

Return type `osid.assessment.AssessmentSection`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Bank

class `dlkit.assessment.objects.`**`Bank`**(*abc_assessment_objects.Bank*, *osid_objects.OsidCatalog*)
`:noindex:`

`get_bank_record`(*bank_record_type*)

Gets the bank record corresponding to the given `Bank` record `Type`.

This method is used to retrieve an object implementing the requested record. The `bank_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(bank_record_type)` is `true` .

Parameters **`bank_record_type`** (`osid.type.Type`) – a bank record type

Returns the bank record

Return type `osid.assessment.records.BankRecord`

Raise `NullArgument` – `bank_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(bank_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Bank Form

class `dlkit.assessment.objects.`**`BankForm`**

Bases: *`dlkit.osid.objects.OsidCatalogForm`*

This is the form for creating and updating banks.

Like all `OsidForm` objects, various data elements may be set here for use in the create and update methods in the `BankAdminSession`. For each data element that may be set, metadata may be examined to provide display hints or data constraints.

get_bank_form_record(*bank_record_type*)
Gets the `BankFormRecord` corresponding to the given bank record `Type`.

Parameters **bank_record_type** (`osid.type.Type`) – a bank record type

Returns the bank record

Return type `osid.assessment.records.BankFormRecord`

Raise `NullArgument` – `bank_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(bank_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Bank List

class `dlkit.assessment.objects.`**BankList**
Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists, BankList` provides a means for accessing `Bank` elements sequentially either one at a time or many at a time.

Examples: while (bl.hasNext()) { Bank bank = bl.getNextBank(); }

or

while (bl.hasNext()) { Bank[] banks = bl.getNextBanks(bl.available());

}

next_bank
Gets the next `Bank` in this list.

Returns the next `Bank` in this list. The `has_next()` method should be used to test that a next `Bank` is available before calling this method.

Return type `osid.assessment.Bank`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

get_next_banks(*n*)
Gets the next set of `Bank` elements in this list which must be less than or equal to the return from `available()`.

Parameters **n** (`cardinal`) – the number of `Bank` elements requested which must be less than or equal to `available()`

Returns an array of `Bank` elements.The length of the array is less than or equal to the number specified.

Return type `osid.assessment.Bank`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Bank Node

class `dlkit.assessment.objects.`**`BankNode`**

Bases: *`dlkit.osid.objects.OsidNode`*

This interface is a container for a partial hierarchy retrieval.

The number of hierarchy levels traversable through this interface depend on the number of levels requested in the `BankHierarchySession`.

`bank`

Gets the `Bank` at this node.

Returns the bank represented by this node

Return type `osid.assessment.Bank`

compliance: mandatory – This method must be implemented.

`parent_bank_nodes`

Gets the parents of this bank.

Returns the parents of this node

Return type `osid.assessment.BankNodeList`

compliance: mandatory – This method must be implemented.

`child_bank_nodes`

Gets the children of this bank.

Returns the children of this node

Return type `osid.assessment.BankNodeList`

compliance: mandatory – This method must be implemented.

Bank Node List

class `dlkit.assessment.objects.`**`BankNodeList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists, BankNodeList` provides a means for accessing `BankNode` elements sequentially either one at a time or many at a time.

Examples: while (bnl.hasNext()) { BankNode node = bnl.getNextBankNode(); }

or

while (bnl.hasNext()) { BankNode[] nodes = bnl.getNextBankNodes(bnl.available());

}

`next_bank_node`

Gets the next `BankNode` in this list.

Returns the next `BankNode` in this list. The `has_next()` method should be used to test that a next `BankNode` is available before calling this method.

Return type `osid.assessment.BankNode`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_bank_nodes`(*n*)

Gets the next set of `BankNode` elements in this list which must be less than or equal to the return from `available()`.

Parameters **n** (`cardinal`) – the number of `BankNode` elements requested which must be less than or equal to `available()`

Returns an array of `BanklNode` elements.The length of the array is less than or equal to the number specified.

Return type `osid.assessment.BankNode`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Response List

class `dlkit.assessment.objects.`**`ResponseList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists, ResponseList` provides a means for accessing `Response` elements sequentially either one at a time or many at a time.

Examples: while (rl.hasNext()) { Response response = rl.getNextResponse(); }

or

while (rl.hasNext()) { Response[] responses = rl.getNextResponses(rl.available());

}

`next_response`

Gets the next `Response` in this list.

Returns the next `Response` in this list. The `has_next()` method should be used to test that a next `Response` is available before calling this method.

Return type `osid.assessment.Response`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_responses`(*n*)

Gets the next set of `Response` elements in this list which must be less than or equal to the return from `available()`.

Parameters **n** (`cardinal`) – the number of `Response` elements requested which must be less than or equal to `available()`

Returns an array of `Response` elements.The length of the array is less than or equal to the number specified.

Return type `osid.assessment.Response`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Queries

Question Query

class `dlkit.assessment.queries.`**`QuestionQuery`**

Bases: *`dlkit.osid.queries.OsidObjectQuery`*

This is the query for searching questions.

Each method match request produces an `AND` term while multiple invocations of a method produces a nested `OR`.

`get_question_query_record``(`*question_record_type*`)`

Gets the question record query corresponding to the given `Item` record `Type`.

Multiple retrievals produce a nested `OR` term.

Parameters **`question_record_type`** (`osid.type.Type`) – a question record type

Returns the question query record

Return type `osid.assessment.records.QuestionQueryRecord`

Raise `NullArgument` – `question_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(question_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Answer Query

class `dlkit.assessment.queries.`**`AnswerQuery`**

Bases: *`dlkit.osid.queries.OsidObjectQuery`*

This is the query for searching answers.

Each method match request produces an `AND` term while multiple invocations of a method produces a nested `OR`.

`get_answer_query_record``(`*answer_record_type*`)`

Gets the answer record query corresponding to the given `Answer` record `Type`.

Multiple retrievals produce a nested `OR` term.

Parameters **`answer_record_type`** (`osid.type.Type`) – an answer record type

Returns the answer query record

Return type `osid.assessment.records.AnswerQueryRecord`

Raise `NullArgument` – `answer_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(answer_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Item Query

class `dlkit.assessment.queries.`**`ItemQuery`**

Bases: *`dlkit.osid.queries.OsidObjectQuery`*, *`dlkit.osid.queries.OsidAggregateableQuery`*

This is the query for searching items.

Each method match request produces an `AND` term while multiple invocations of a method produces a nested `OR`.

`match_learning_objective_id`(*objective_id*, *match*)

Sets the learning objective `Id` for this query.

Parameters

- **`objective_id`** (`osid.id.Id`) – a learning objective `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for negative match

Raise `NullArgument` – `objective_id` is `null`

compliance: mandatory – This method must be implemented.

`learning_objective_id_terms`

`supports_learning_objective_query`()

Tests if an `ObjectiveQuery` is available.

Returns `true` if a learning objective query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`learning_objective_query`

Gets the query for a learning objective.

Multiple retrievals produce a nested `OR` term.

Returns the learning objective query

Return type `osid.learning.ObjectiveQuery`

Raise `Unimplemented` – `supports_learning_objective_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_learning_objective_query()‘‘ is ‘‘true‘‘.

`match_any_learning_objective`(*match*)

Matches an item with any objective.

Parameters **`match`** (`boolean`) – `true` to match items with any learning objective, `false` to match items with no learning objectives

compliance: mandatory – This method must be implemented.

`learning_objective_terms`

`match_question_id`(*question_id*, *match*)

Sets the question `Id` for this query.

Parameters

- **`question_id`** (`osid.id.Id`) – a question `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `question_id` is `null`

compliance: mandatory – This method must be implemented.

`question_id_terms`

`supports_question_query`()
Tests if a `QuestionQuery` is available.

Returns `true` if a question query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`question_query`
Gets the query for a question.

Multiple retrievals produce a nested `OR` term.

Returns the question query

Return type `osid.assessment.QuestionQuery`

Raise `Unimplemented` – `supports_question_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_learning_objective_query()‘‘ is ‘‘true‘‘.

`match_any_question`(*match*)
Matches an item with any question.

Parameters **`match`** (`boolean`) – `true` to match items with any question, `false` to match items with no questions

compliance: mandatory – This method must be implemented.

`question_terms`

`match_answer_id`(*answer_id*, *match*)
Sets the answer `Id` for this query.

Parameters

- **`answer_id`** (`osid.id.Id`) – an answer `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `answer_id` is `null`

compliance: mandatory – This method must be implemented.

`answer_id_terms`

`supports_answer_query`()
Tests if an `AnswerQuery` is available.

Returns `true` if an answer query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`answer_query`
Gets the query for an answer.

Multiple retrievals produce a nested `OR` term.

Returns the answer query

Return type `osid.assessment.AnswerQuery`

Raise `Unimplemented` – `supports_answer_query()` is `false`

compliance: optional – This method must be implemented if ``supports_learning_objective_query()`` is ``true``.

match_any_answer(*match*)

Matches an item with any answer.

Parameters match (`boolean`) – `true` to match items with any answer, `false` to match items with no answers

compliance: mandatory – This method must be implemented.

answer_terms

match_assessment_id(*assessment_id*, *match*)

Sets the assessment `Id` for this query.

Parameters

- **assessment_id** (`osid.id.Id`) – an assessment `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for negative match

Raise `NullArgument` – `assessment_id` is `null`

compliance: mandatory – This method must be implemented.

assessment_id_terms

supports_assessment_query()

Tests if an `AssessmentQuery` is available.

Returns `true` if an assessment query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

assessment_query

Gets the query for an assessment.

Multiple retrievals produce a nested `OR` term.

Returns the assessment query

Return type `osid.assessment.AssessmentQuery`

Raise `Unimplemented` – `supports_assessment_query()` is `false`

compliance: optional – This method must be implemented if ``supports_assessment_query()`` is ``true``.

match_any_assessment(*match*)

Matches an item with any assessment.

Parameters match (`boolean`) – `true` to match items with any assessment, `false` to match items with no assessments

compliance: mandatory – This method must be implemented.

assessment_terms

match_bank_id(*bank_id*, *match*)

Sets the bank `Id` for this query.

Parameters

- **bank_id** (osid.id.Id) – a bank Id
- **match** (boolean) – true for a positive match, false for negative match

Raise NullArgument – bank_id is null

compliance: mandatory – This method must be implemented.

bank_id_terms

supports_bank_query()

Tests if a BankQuery is available.

Returns true if a bank query is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

bank_query

Gets the query for a bank.

Multiple retrievals produce a nested OR term.

Returns the bank query

Return type osid.assessment.BankQuery

Raise Unimplemented – supports_bank_query() is false

compliance: optional – This method must be implemented if ‘‘supports_bank_query()‘‘ is ‘‘true‘‘.

bank_terms

get_item_query_record(*item_record_type*)

Gets the item record query corresponding to the given Item record Type.

Multiple retrievals produce a nested OR term.

Parameters **item_record_type** (osid.type.Type) – an item record type

Returns the item query record

Return type osid.assessment.records.ItemQueryRecord

Raise NullArgument – item_record_type is null

Raise OperationFailed – unable to complete request

Raise Unsupported – has_record_type(item_record_type) is false

compliance: mandatory – This method must be implemented.

Assessment Query

class dlkit.assessment.queries.**AssessmentQuery**

Bases: *dlkit.osid.queries.OsidObjectQuery*

This is the query for searching assessments.

Each method match request produces an AND term while multiple invocations of a method produces a nested OR.

match_level_id(*grade_id*, *match*)

Sets the level grade Id for this query.

Parameters

- **grade_id** (osid.id.Id) – a grade Id
- **match** (boolean) – true for a positive match, false for a negative match

Raise NullArgument – grade_id is null

compliance: mandatory – This method must be implemented.

level_id_terms

supports_level_query()

Tests if a GradeQuery is available.

Returns true if a grade query is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

level_query

Gets the query for a grade.

Multiple retrievals produce a nested OR term.

Returns the grade query

Return type osid.grading.GradeQuery

Raise Unimplemented – supports_level_query() is false

compliance: optional – This method must be implemented if ``supports_level_query()`` is ``true``.

match_any_level(*match*)

Matches an assessment that has any level assigned.

Parameters match (boolean) – true to match assessments with any level, false to match assessments with no level

compliance: mandatory – This method must be implemented.

level_terms

match_rubric_id(*assessment_id*, *match*)

Sets the rubric assessment Id for this query.

Parameters

- **assessment_id** (osid.id.Id) – an assessment Id
- **match** (boolean) – true for a positive match, false for a negative match

Raise NullArgument – assessment_id is null

compliance: mandatory – This method must be implemented.

rubric_id_terms

supports_rubric_query()

Tests if an AssessmentQuery is available.

Returns true if a rubric assessment query is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

rubric_query

Gets the query for a rubric assessment.

Multiple retrievals produce a nested `OR` term.

Returns the assessment query

Return type `osid.assessment.AssessmentQuery`

Raise `Unimplemented` – `supports_rubric_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_rubric_query()‘‘ is ‘‘true‘‘.

match_any_rubric(*match*)

Matches an assessment that has any rubric assessment assigned.

Parameters match (`boolean`) – `true` to match assessments with any rubric, `false` to match assessments with no rubric

compliance: mandatory – This method must be implemented.

rubric_terms

match_item_id(*item_id*, *match*)

Sets the item `Id` for this query.

Parameters

- **item_id** (`osid.id.Id`) – an item `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `item_id` is `null`

compliance: mandatory – This method must be implemented.

item_id_terms

supports_item_query()

Tests if an `ItemQuery` is available.

Returns `true` if an item query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

item_query

Gets the query for an item.

Multiple retrievals produce a nested `OR` term.

Returns the item query

Return type `osid.assessment.ItemQuery`

Raise `Unimplemented` – `supports_item_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_item_query()‘‘ is ‘‘true‘‘.

match_any_item(*match*)

Matches an assessment that has any item.

Parameters match (`boolean`) – `true` to match assessments with any item, `false` to match assessments with no items

compliance: mandatory – This method must be implemented.

item_terms

match_assessment_offered_id(*assessment_offered_id*, *match*)
Sets the assessment offered `Id` for this query.

Parameters

- **assessment_offered_id** (`osid.id.Id`) – an assessment offered `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `assessment_offered_id` is `null`

compliance: mandatory – This method must be implemented.

assessment_offered_id_terms

supports_assessment_offered_query()
Tests if an `AssessmentOfferedQuery` is available.

Returns `true` if an assessment offered query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

assessment_offered_query
Gets the query for an assessment offered.

Multiple retrievals produce a nested `OR` term.

Returns the assessment offered query

Return type `osid.assessment.AssessmentOfferedQuery`

Raise `Unimplemented` – `supports_assessment_offered_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_assessment_offered_query()‘‘ is ‘‘true‘‘.

match_any_assessment_offered(*match*)
Matches an assessment that has any offering.

Parameters match (`boolean`) – `true` to match assessments with any offering, `false` to match assessments with no offerings

compliance: mandatory – This method must be implemented.

assessment_offered_terms

match_assessment_taken_id(*assessment_taken_id*, *match*)
Sets the assessment taken `Id` for this query.

Parameters

- **assessment_taken_id** (`osid.id.Id`) – an assessment taken `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `assessment_taken_id` is `null`

compliance: mandatory – This method must be implemented.

assessment_taken_id_terms

supports_assessment_taken_query()
Tests if an `AssessmentTakenQuery` is available.

Returns `true` if an assessment taken query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`assessment_taken_query`
Gets the query for an assessment taken.

Multiple retrievals produce a nested `OR` term.

Returns the assessment taken query

Return type `osid.assessment.AssessmentTakenQuery`

Raise `Unimplemented` – `supports_assessment_taken_query()` is `false`

compliance: optional – This method must be implemented if ``supports_assessment_taken_query()`` is ``true``.

`match_any_assessment_taken`(*match*)
Matches an assessment that has any taken version.

Parameters **`match`** (`boolean`) – `true` to match assessments with any taken assessments, `false` to match assessments with no taken assessments

compliance: mandatory – This method must be implemented.

`assessment_taken_terms`

`match_bank_id`(*bank_id*, *match*)
Sets the bank `Id` for this query.

Parameters

- **`bank_id`** (`osid.id.Id`) – a bank `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `bank_id` is `null`

compliance: mandatory – This method must be implemented.

`bank_id_terms`

`supports_bank_query`()
Tests if a `BankQuery` is available.

Returns `true` if a bank query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`bank_query`
Gets the query for a bank.

Multiple retrievals produce a nested `OR` term.

Returns the bank query

Return type `osid.assessment.BankQuery`

Raise `Unimplemented` – `supports_bank_query()` is `false`

compliance: optional – This method must be implemented if ``supports_bank_query()`` is ``true``.

`bank_terms`

`get_assessment_query_record`(*assessment_record_type*)
Gets the assessment query record corresponding to the given `Assessment` record `Type`.

Multiple retrievals produce a nested `OR` term.

Parameters **`assessment_record_type`** (`osid.type.Type`) – an assessment record type

Returns the assessment query record

Return type `osid.assessment.records.AssessmentQueryRecord`

Raise `NullArgument` – `assessment_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(assessment_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Assessment Offered Query

class `dlkit.assessment.queries.`**`AssessmentOfferedQuery`**

Bases: *`dlkit.osid.queries.OsidObjectQuery`*, *`dlkit.osid.queries.OsidSubjugateableQuery`*

This is the query for searching assessments.

Each method match request produces an `AND` term while multiple invocations of a method produces a nested `OR`.

`match_assessment_id`(*assessment_id*, *match*)

Sets the assessment `Id` for this query.

Parameters

- **`assessment_id`** (`osid.id.Id`) – an assessment `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `assessment_id` is `null`

compliance: mandatory – This method must be implemented.

`assessment_id_terms`

`supports_assessment_query`()

Tests if an `AssessmentQuery` is available.

Returns `true` if an assessment query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`assessment_query`

Gets the query for an assessment.

Multiple retrievals produce a nested `OR` term.

Returns the assessment query

Return type `osid.assessment.AssessmentQuery`

Raise `Unimplemented` – `supports_assessment_query()` is `false`

compliance: optional – This method must be implemented if ``supports_assessment_query()`` is ``true``.

`assessment_terms`

`match_level_id`(*grade_id*, *match*)

Sets the level grade `Id` for this query.

Parameters

- **grade_id** (`osid.id.Id`) – a grade `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `grade_id` is `null`

compliance: mandatory – This method must be implemented.

level_id_terms

supports_level_query()
Tests if a `GradeQuery` is available.

Returns `true` if a grade query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

level_query
Gets the query for a grade.

Multiple retrievals produce a nested `OR` term.

Returns the grade query

Return type `osid.grading.GradeQuery`

Raise `Unimplemented` – `supports_level_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_level_query()‘‘ is ‘‘true‘‘.

match_any_level(*match*)
Matches an assessment offered that has any level assigned.

Parameters **match** (`boolean`) – `true` to match offerings with any level, `false` to match offerings with no levsls

compliance: mandatory – This method must be implemented.

level_terms

match_items_sequential(*match*)
Match sequential assessments.

Parameters **match** (`boolean`) – `true` for a positive match, `false` for a negative match

compliance: mandatory – This method must be implemented.

items_sequential_terms

match_items_shuffled(*match*)
Match shuffled item assessments.

Parameters **match** (`boolean`) – `true` for a positive match, `false` for a negative match

compliance: mandatory – This method must be implemented.

items_shuffled_terms

match_start_time(*start*, *end*, *match*)
Matches assessments whose start time falls between the specified range inclusive.

Parameters

- **start** (`osid.calendaring.DateTime`) – start of range

- **end** (`osid.calendaring.DateTime`) – end of range
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `end` is less than `start`

compliance: mandatory – This method must be implemented.

match_any_start_time (*match*)

Matches offerings that has any start time assigned.

Parameters **match** (`boolean`) – `true` to match offerings with any start time, `false` to match offerings with no start time

compliance: mandatory – This method must be implemented.

start_time_terms

match_deadline (*start*, *end*, *match*)

Matches assessments whose end time falls between the specified range inclusive.

Parameters

- **start** (`osid.calendaring.DateTime`) – start of range
- **end** (`osid.calendaring.DateTime`) – end of range
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `end` is less than `start`

Raise `NullArgument` – `start` or `end` is `null`

compliance: mandatory – This method must be implemented.

match_any_deadline (*match*)

Matches offerings that have any deadline assigned.

Parameters **match** (`boolean`) – `true` to match offerings with any deadline, `false` to match offerings with no deadline

compliance: mandatory – This method must be implemented.

deadline_terms

match_duration (*low*, *high*, *match*)

Matches assessments whose duration falls between the specified range inclusive.

Parameters

- **low** (`osid.calendaring.Duration`) – start range of duration
- **high** (`osid.calendaring.Duration`) – end range of duration
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `end` is less than `start`

Raise `NullArgument` – `start` or `end` is `null`

compliance: mandatory – This method must be implemented.

match_any_duration (*match*)

Matches offerings that have any duration assigned.

Parameters **match** (`boolean`) – `true` to match offerings with any duration, `false` to match offerings with no duration

compliance: mandatory – This method must be implemented.

duration_terms

match_score_system_id(*grade_system_id*, *match*)

Sets the grade system `Id` for this query.

Parameters

- **grade_system_id** (`osid.id.Id`) – a grade system `Id`
- **match** (boolean) – `true for a positive match, false for a negative match`

Raise `NullArgument` – `grade_system_id` is `null`

compliance: mandatory – This method must be implemented.

score_system_id_terms

supports_score_system_query()

Tests if a `GradeSystemQuery` is available.

Returns `true` if a grade system query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

score_system_query

Gets the query for a grade system.

Multiple retrievals produce a nested `OR` term.

Returns the grade system query

Return type `osid.grading.GradeSystemQuery`

Raise `Unimplemented` – `supports_score_system_query()` is `false`

compliance: optional – This method must be implemented if ``supports_score_system_query()`` is ``true``.

match_any_score_system(*match*)

Matches taken assessments that have any grade system assigned.

Parameters match (`boolean`) – `true` to match assessments with any grade system, `false` to match assessments with no grade system

compliance: mandatory – This method must be implemented.

score_system_terms

match_grade_system_id(*grade_system_id*, *match*)

Sets the grade system `Id` for this query.

Parameters

- **grade_system_id** (`osid.id.Id`) – a grade system `Id`
- **match** (boolean) – `true for a positive match, false for a negative match`

Raise `NullArgument` – `grade_system_id` is `null`

compliance: mandatory – This method must be implemented.

grade_system_id_terms

supports_grade_system_query()

Tests if a `GradeSystemQuery` is available.

Returns `true` if a grade system query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`grade_system_query`

Gets the query for a grade system.

Multiple retrievals produce a nested `OR` term.

Returns the grade system query

Return type `osid.grading.GradeSystemQuery`

Raise `Unimplemented` – `supports_score_system_query()` is `false`

compliance: optional – This method must be implemented if ``supports_score_system_query()`` is ``true``.

`match_any_grade_system`(*match*)

Matches taken assessments that have any grade system assigned.

Parameters **`match`** (`boolean`) – `true` to match assessments with any grade system, `false` to match assessments with no grade system

compliance: mandatory – This method must be implemented.

`grade_system_terms`

`match_rubric_id`(*assessment_offered_id*, *match*)

Sets the rubric assessment offered `Id` for this query.

Parameters

- **`assessment_offered_id`** (`osid.id.Id`) – an assessment offered `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `assessment_offered_id` is `null`

compliance: mandatory – This method must be implemented.

`rubric_id_terms`

`supports_rubric_query`()

Tests if an `AssessmentOfferedQuery` is available.

Returns `true` if a rubric assessment offered query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`rubric_query`

Gets the query for a rubric assessment.

Multiple retrievals produce a nested `OR` term.

Returns the assessment offered query

Return type `osid.assessment.AssessmentOfferedQuery`

Raise `Unimplemented` – `supports_rubric_query()` is `false`

compliance: optional – This method must be implemented if ``supports_rubric_query()`` is ``true``.

`match_any_rubric`(*match*)

Matches an assessment offered that has any rubric assessment assigned.

Parameters **match** (boolean) – true to match assessments offered with any rubric, false to match assessments offered with no rubric

compliance: mandatory – This method must be implemented.

rubric_terms

match_assessment_taken_id(*assessment_taken_id*, *match*)

Sets the assessment taken Id for this query.

Parameters

- **assessment_taken_id** (osid.id.Id) – an assessment taken Id
- **match** (boolean) – true for a positive match, false for a negative match

Raise NullArgument – assessment_taken_id is null

compliance: mandatory – This method must be implemented.

assessment_taken_id_terms

supports_assessment_taken_query()

Tests if an AssessmentTakenQuery is available.

Returns true if an assessment taken query is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

assessment_taken_query

Gets the query for an assessment taken.

Multiple retrievals produce a nested OR term.

Returns the assessment taken query

Return type osid.assessment.AssessmentTakenQuery

Raise Unimplemented – supports_assessment_taken_query() is false

compliance: optional – This method must be implemented if ``supports_assessment_taken_query()`` is ``true``.

match_any_assessment_taken(*match*)

Matches offerings that have any taken assessment version.

Parameters **match** (boolean) – true to match offerings with any taken assessment, false to match offerings with no assessmen taken

compliance: mandatory – This method must be implemented.

assessment_taken_terms

match_bank_id(*bank_id*, *match*)

Sets the bank Id for this query.

Parameters

- **bank_id** (osid.id.Id) – a bank Id
- **match** (boolean) – true for a positive match, false for a negative match

Raise NullArgument – bank_id is null

compliance: mandatory – This method must be implemented.

bank_id_terms

supports_bank_query()

Tests if a BankQuery is available.

Returns true if a bank query is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

bank_query

Gets the query for a bank.

Multiple retrievals produce a nested OR term.

Returns the bank query

Return type osid.assessment.BankQuery

Raise Unimplemented – supports_bank_query() is false

compliance: optional – This method must be implemented if ``supports_bank_query()`` is ``true``.

bank_terms

get_assessment_offered_query_record(*assessment_offered_record_type*)

Gets the assessment offered query record corresponding to the given AssessmentOffered record Type.

Multiple retrievals produce a nested OR term.

Parameters **assessment_offered_record_type** (osid.type.Type) – an assessment offered record type

Returns the assessment offered query record

Return type osid.assessment.records.AssessmentOfferedQueryRecord

Raise NullArgument – assessment_offered_record_type is null

Raise OperationFailed – unable to complete request

Raise Unsupported – has_record_type(assessment_offered_record_type) is false

compliance: mandatory – This method must be implemented.

Assessment Taken Query

class dlkit.assessment.queries.**AssessmentTakenQuery**

Bases: *dlkit.osid.queries.OsidObjectQuery*

This is the query for searching assessments.

Each method match request produces an AND term while multiple invocations of a method produces a nested OR.

match_assessment_offered_id(*assessment_offered_id, match*)

Sets the assessment offered Id for this query.

Parameters

- **assessment_offered_id** (osid.id.Id) – an assessment Id
- **match** (boolean) – true for a positive match, false for a negative match

Raise NullArgument – assessment_offered_id is null

compliance: mandatory – This method must be implemented.

assessment_offered_id_terms

supports_assessment_offered_query()
Tests if an AssessmentOfferedQuery is available.

Returns true if an assessment offered query is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

assessment_offered_query
Gets the query for an assessment.

Multiple retrievals produce a nested OR term.

Returns the assessment offered query

Return type osid.assessment.AssessmentOfferedQuery

Raise Unimplemented – supports_assessment_offered_query() is false

compliance: optional – This method must be implemented if ``supports_assessment_offered_query()`` is ``true``.

assessment_offered_terms

match_taker_id(*resource_id*, *match*)
Sets the resource Id for this query.

Parameters

- **resource_id** (osid.id.Id) – a resource Id
- **match** (boolean) – true for a positive match, false for a negative match

Raise NullArgument – resource_id is null

compliance: mandatory – This method must be implemented.

taker_id_terms

supports_taker_query()
Tests if a ResourceQuery is available.

Returns true if a resource query is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

taker_query
Gets the query for a resource.

Multiple retrievals produce a nested OR term.

Returns the resource query

Return type osid.resource.ResourceQuery

Raise Unimplemented – supports_taker_query() is false

compliance: optional – This method must be implemented if ``supports_taker_query()`` is ``true``.

taker_terms

match_taking_agent_id(*agent_id*, *match*)
Sets the agent `Id` for this query.

Parameters

- **agent_id** (`osid.id.Id`) – an agent `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `agent_id` is `null`

compliance: mandatory – This method must be implemented.

taking_agent_id_terms

supports_taking_agent_query()
Tests if an `AgentQuery` is available.

Returns `true` if an agent query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

taking_agent_query
Gets the query for an agent.

Multiple retrievals produce a nested `OR` term.

Returns the agent query

Return type `osid.authentication.AgentQuery`

Raise `Unimplemented` – `supports_taking_agent_query()` is `false`

compliance: optional – This method must be implemented if ``supports_taking_agent_query()`` is ``true``.

taking_agent_terms

match_actual_start_time(*start*, *end*, *match*)
Matches assessments whose start time falls between the specified range inclusive.

Parameters

- **start** (`osid.calendaring.DateTime`) – start of range
- **end** (`osid.calendaring.DateTime`) – end of range
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `end` is less than `start`

Raise `NullArgument` – `start` or `end` is `null`

compliance: mandatory – This method must be implemented.

match_any_actual_start_time(*match*)
Matches taken assessments taken that have begun.

Parameters match (`boolean`) – `true` to match assessments taken started, `false` to match assessments taken that have not begun

compliance: mandatory – This method must be implemented.

actual_start_time_terms

match_completion_time(*start*, *end*, *match*)
Matches assessments whose completion time falls between the specified range inclusive.

Parameters

- **start** (`osid.calendaring.DateTime`) – start of range
- **end** (`osid.calendaring.DateTime`) – end of range
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `end` is less than `start`

Raise `NullArgument` – `start` or `end` is `null`

compliance: mandatory – This method must be implemented.

match_any_completion_time(*match*)

Matches taken assessments taken that have completed.

Parameters match (`boolean`) – `true` to match assessments taken completed, `false` to match assessments taken that are incomplete

compliance: mandatory – This method must be implemented.

completion_time_terms

match_time_spent(*low*, *high*, *match*)

Matches assessments where the time spent falls between the specified range inclusive.

Parameters

- **low** (`osid.calendaring.Duration`) – start of duration range
- **high** (`osid.calendaring.Duration`) – end of duration range
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `high` is less than `low`

Raise `NullArgument` – `low` or `high` is `null`

compliance: mandatory – This method must be implemented.

time_spent_terms

match_score_system_id(*grade_system_id*, *match*)

Sets the grade system `Id` for this query.

Parameters

- **grade_system_id** (`osid.id.Id`) – a grade system `Id`
- **match** (`boolean`) – `true for a positive match, false for a negative match`

Raise `NullArgument` – `grade_system_id` is `null`

compliance: mandatory – This method must be implemented.

score_system_id_terms

supports_score_system_query()

Tests if a `GradeSystemQuery` is available.

Returns `true` if a grade system query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

score_system_query

Gets the query for a grade system.

Multiple retrievals produce a nested `OR` term.

Returns the grade system query

Return type `osid.grading.GradeSystemQuery`

Raise `Unimplemented` – `supports_score_system_query()` is `false`

compliance: optional – This method must be implemented if ``supports_score_system_query()`` is ``true``.

match_any_score_system(*match*)

Matches taken assessments that have any grade system assigned.

Parameters **match** (`boolean`) – `true` to match assessments with any grade system, `false` to match assessments with no grade system

compliance: mandatory – This method must be implemented.

score_system_terms

match_score(*low*, *high*, *match*)

Matches assessments whose score falls between the specified range inclusive.

Parameters

- **low** (`decimal`) – start of range
- **high** (`decimal`) – end of range
- **match** (`boolean`) – `true` for a positive match, `false` for negative match

Raise `InvalidArgument` – `high` is less than `low`

compliance: mandatory – This method must be implemented.

match_any_score(*match*)

Matches taken assessments that have any score assigned.

Parameters **match** (`boolean`) – `true` to match assessments with any score, `false` to match assessments with no score

compliance: mandatory – This method must be implemented.

score_terms

match_grade_id(*grade_id*, *match*)

Sets the grade `Id` for this query.

Parameters

- **grade_id** (`osid.id.Id`) – a grade `Id`
- **match** (boolean) – `true for a positive match, false for a negative match`

Raise `NullArgument` – `grade_id` is `null`

compliance: mandatory – This method must be implemented.

grade_id_terms

supports_grade_query()

Tests if a `GradeQuery` is available.

Returns `true` if a grade query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`grade_query`

Gets the query for a grade.

Multiple retrievals produce a nested `OR` term.

Returns the grade query

Return type `osid.grading.GradeQuery`

Raise `Unimplemented` – `supports_grade_query()` is `false`

compliance: optional – This method must be implemented if ``supports_grade_query()`` is ``true``.

`match_any_grade`(*match*)

Matches taken assessments that have any grade assigned.

Parameters **`match`** (`boolean`) – `true` to match assessments with any grade, `false` to match assessments with no grade

compliance: mandatory – This method must be implemented.

`grade_terms`

`match_feedback`(*comments*, *string_match_type*, *match*)

Sets the comment string for this query.

Parameters

- **`comments`** (`string`) – comment string
- **`string_match_type`** (`osid.type.Type`) – the string match type
- **`match`** (`boolean`) – `true` for a positive match, `false` for negative match

Raise `InvalidArgument` – `comments is` not of `string_match_type`

Raise `NullArgument` – `comments` or `string_match_type` is `null`

Raise `Unsupported` – `supports_string_match_type(string_match_type)` is `false`

compliance: mandatory – This method must be implemented.

`match_any_feedback`(*match*)

Matches taken assessments that have any comments.

Parameters **`match`** (`boolean`) – `true` to match assessments with any comments, `false` to match assessments with no comments

compliance: mandatory – This method must be implemented.

`feedback_terms`

`match_rubric_id`(*assessment_taken_id*, *match*)

Sets the rubric assessment taken `Id` for this query.

Parameters

- **`assessment_taken_id`** (`osid.id.Id`) – an assessment taken `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `assessment_taken_id` is `null`

compliance: mandatory – This method must be implemented.

rubric_id_terms

supports_rubric_query()
Tests if an `AssessmentTakenQuery` is available.

Returns `true` if a rubric assessment taken query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

rubric_query
Gets the query for a rubric assessment.

Multiple retrievals produce a nested `OR` term.

Returns the assessment taken query

Return type `osid.assessment.AssessmentTakenQuery`

Raise `Unimplemented` – `supports_rubric_query()` is `false`

compliance: optional – This method must be implemented if ``supports_rubric_query()`` is ``true``.

match_any_rubric(*match*)
Matches an assessment taken that has any rubric assessment assigned.

Parameters **match** (`boolean`) – `true` to match assessments taken with any rubric, `false` to match assessments taken with no rubric

compliance: mandatory – This method must be implemented.

rubric_terms

match_bank_id(*bank_id*, *match*)
Sets the bank `Id` for this query.

Parameters

- **bank_id** (`osid.id.Id`) – a bank `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `bank_id` is `null`

compliance: mandatory – This method must be implemented.

bank_id_terms

supports_bank_query()
Tests if a `BankQuery` is available.

Returns `true` if a bank query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

bank_query
Gets the query for a bank.

Multiple retrievals produce a nested `OR` term.

Returns the bank query

Return type `osid.assessment.BankQuery`

Raise `Unimplemented` – `supports_bank_query()` is `false` *compliance: optional – This method must be implemented if ‘‘supports_bank_query()‘‘ is ‘‘true‘‘.*

bank_terms

get_assessment_taken_query_record(*assessment_taken_record_type*)
Gets the assessment taken query record corresponding to the given `AssessmentTaken` record `Type`.

Multiple retrievals produce a nested `OR` term.

Parameters **assessment_taken_record_type** (`osid.type.Type`) – an assessment taken record type

Returns the assessment taken query record

Return type `osid.assessment.records.AssessmentTakenQueryRecord`

Raise `NullArgument` – `assessment_taken_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(assessment_taken_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Bank Query

class `dlkit.assessment.queries.`**BankQuery**
Bases: *`dlkit.osid.queries.OsidCatalogQuery`*

This is the query for searching banks Each method specifies an `AND` term while multiple invocations of the same method produce a nested `OR`.

match_item_id(*item_id, match*)
Sets the item `Id` for this query.

Parameters

- **item_id** (`osid.id.Id`) – an item `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `item_id` is `null`

compliance: mandatory – This method must be implemented.

item_id_terms

supports_item_query()
Tests if a `ItemQuery` is available.

Returns `true` if an item query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

item_query
Gets the query for an item.

Multiple retrievals produce a nested `OR` term.

Returns the item query

Return type `osid.assessment.ItemQuery`

Raise `Unimplemented` – `supports_item_query()` is `false` *compliance: optional – This method must be implemented if ``supports_item_query()`` is ``true``.*

`match_any_item`(*match*)

Matches assessment banks that have any item assigned.

Parameters **`match`** (`boolean`) – `true` to match banks with any item, `false` to match assessments with no item

compliance: mandatory – This method must be implemented.

`item_terms`

`match_assessment_id`(*assessment_id*, *match*)

Sets the assessment `Id` for this query.

Parameters

- **`assessment_id`** (`osid.id.Id`) – an assessment `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `assessment_id` is `null`

compliance: mandatory – This method must be implemented.

`assessment_id_terms`

`supports_assessment_query`()

Tests if an `AssessmentQuery` is available.

Returns `true` if an assessment query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`assessment_query`

Gets the query for an assessment.

Multiple retrievals produce a nested `OR` term.

Returns the assessment query

Return type `osid.assessment.AssessmentQuery`

Raise `Unimplemented` – `supports_assessment_query()` is `false`

compliance: optional – This method must be implemented if ``supports_assessment_query()`` is ``true``.

`match_any_assessment`(*match*)

Matches assessment banks that have any assessment assigned.

Parameters **`match`** (`boolean`) – `true` to match banks with any assessment, `false` to match banks with no assessment

compliance: mandatory – This method must be implemented.

`assessment_terms`

`match_assessment_offered_id`(*assessment_offered_id*, *match*)

Sets the assessment offered `Id` for this query.

Parameters

- **`assessment_offered_id`** (`osid.id.Id`) – an assessment `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `assessment_offered_id` is `null`

compliance: mandatory – This method must be implemented.

assessment_offered_id_terms

supports_assessment_offered_query()
Tests if an `AssessmentOfferedQuery` is available.

Returns `true` if an assessment offered query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

assessment_offered_query
Gets the query for an assessment offered.

Multiple retrievals produce a nested `OR` term.

Returns the assessment offered query

Return type `osid.assessment.AssessmentOfferedQuery`

Raise `Unimplemented` – `supports_assessment_offered_query()` is `false`

compliance: optional – This method must be implemented if ``supports_assessment_offered_query()`` is ``true``.

match_any_assessment_offered(*match*)
Matches assessment banks that have any assessment offering assigned.

Parameters **match** (`boolean`) – `true` to match banks with any assessment offering, `false` to match banks with no offering

compliance: mandatory – This method must be implemented.

assessment_offered_terms

match_ancestor_bank_id(*bank_id*, *match*)
Sets the bank `Id` for to match banks in which the specified bank is an acestor.

Parameters

- **bank_id** (`osid.id.Id`) – a bank `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `bank_id` is `null`

compliance: mandatory – This method must be implemented.

ancestor_bank_id_terms

supports_ancestor_bank_query()
Tests if a `BankQuery` is available.

Returns `true` if a bank query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

ancestor_bank_query
Gets the query for an ancestor bank.

Multiple retrievals produce a nested `OR` term.

Returns the bank query

Return type `osid.assessment.BankQuery`

Raise `Unimplemented` – `supports_ancestor_bank_query()` is `false`

compliance: optional – This method must be implemented if ``supports_ancestor_bank_query()`` is ``true``.

`match_any_ancestor_bank`(*match*)
Matches a bank that has any ancestor.

Parameters **`match`** (`boolean`) – `true` to match banks with any ancestor banks, `false` to match root banks

compliance: mandatory – This method must be implemented.

`ancestor_bank_terms`

`match_descendant_bank_id`(*bank_id*, *match*)
Sets the bank `Id` for to match banks in which the specified bank is a descendant.

Parameters

- **`bank_id`** (`osid.id.Id`) – a bank `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `bank_id` is `null`

compliance: mandatory – This method must be implemented.

`descendant_bank_id_terms`

`supports_descendant_bank_query`()
Tests if a `BankQuery` is available.

Returns `true` if a bank query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`descendant_bank_query`
Gets the query for a descendant bank.

Multiple retrievals produce a nested `OR` term.

Returns the bank query

Return type `osid.assessment.BankQuery`

Raise `Unimplemented` – `supports_descendant_bank_query()` is `false`

compliance: optional – This method must be implemented if ``supports_descendant_bank_query()`` is ``true``.

`match_any_descendant_bank`(*match*)
Matches a bank that has any descendant.

Parameters **`match`** (`boolean`) – `true` to match banks with any descendant banks, `false` to match leaf banks

compliance: mandatory – This method must be implemented.

`descendant_bank_terms`

`get_bank_query_record`(*bank_record_type*)
Gets the bank query record corresponding to the given `Bank` record `Type`.

Multiple record retrievals produce a nested `OR` term.

Parameters **bank_record_type** (osid.type.Type) – a bank record type

Returns the bank query record

Return type osid.assessment.records.BankQueryRecord

Raise NullArgument – bank_record_type is null

Raise OperationFailed – unable to complete request

Raise Unsupported – has_record_type(bank_record_type) is false

compliance: mandatory – This method must be implemented.

Records

Question Record

class dlkit.assessment.records.**QuestionRecord**

Bases: *dlkit.osid.records.OsidRecord*

A record for a Question.

The methods specified by the record type are available through the underlying object.

Question Query Record

class dlkit.assessment.records.**QuestionQueryRecord**

Bases: *dlkit.osid.records.OsidRecord*

A record for a QuestionQuery.

The methods specified by the record type are available through the underlying object.

Question Form Record

class dlkit.assessment.records.**QuestionFormRecord**

Bases: *dlkit.osid.records.OsidRecord*

A record for a QuestionForm.

The methods specified by the record type are available through the underlying object.

Answer Record

class dlkit.assessment.records.**AnswerRecord**

Bases: *dlkit.osid.records.OsidRecord*

A record for an Answer.

The methods specified by the record type are available through the underlying object.

Answer Query Record

class `dlkit.assessment.records.`**`AnswerQueryRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `AnswerQuery`.

The methods specified by the record type are available through the underlying object.

Answer Form Record

class `dlkit.assessment.records.`**`AnswerFormRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `AnswerForm`.

The methods specified by the record type are available through the underlying object.

Item Record

class `dlkit.assessment.records.`**`ItemRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `Item`.

The methods specified by the record type are available through the underlying object.

Item Query Record

class `dlkit.assessment.records.`**`ItemQueryRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `ItemQuery`.

The methods specified by the record type are available through the underlying object.

Item Form Record

class `dlkit.assessment.records.`**`ItemFormRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `ItemForm`.

The methods specified by the record type are available through the underlying object.

Item Search Record

class `dlkit.assessment.records.`**`ItemSearchRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `ItemSearch`.

The methods specified by the record type are available through the underlying object.

Assessment Record

class `dlkit.assessment.records.`**`AssessmentRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `Assessment`.

The methods specified by the record type are available through the underlying object.

Assessment Query Record

class `dlkit.assessment.records.`**`AssessmentQueryRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `AssessmentQuery`.

The methods specified by the record type are available through the underlying object.

Assessment Form Record

class `dlkit.assessment.records.`**`AssessmentFormRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `AssessmentForm`.

The methods specified by the record type are available through the underlying object.

Assessment Search Record

class `dlkit.assessment.records.`**`AssessmentSearchRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `AssessmentSearch`.

The methods specified by the record type are available through the underlying object.

Assessment Offered Record

class `dlkit.assessment.records.`**`AssessmentOfferedRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `AssessmentOffered`.

The methods specified by the record type are available through the underlying object.

Assessment Offered Query Record

class `dlkit.assessment.records.`**`AssessmentOfferedQueryRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `AssessmentOfferedQuery`.

The methods specified by the record type are available through the underlying object.

Assessment Offered Form Record

class `dlkit.assessment.records.`**`AssessmentOfferedFormRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `AssessmentOfferedForm`.

The methods specified by the record type are available through the underlying object.

Assessment Offered Search Record

class `dlkit.assessment.records.`**`AssessmentOfferedSearchRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `AssessmentOfferedSearch`.

The methods specified by the record type are available through the underlying object.

Assessment Taken Record

class `dlkit.assessment.records.`**`AssessmentTakenRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `AssessmentTaken`.

The methods specified by the record type are available through the underlying object.

Assessment Taken Query Record

class `dlkit.assessment.records.`**`AssessmentTakenQueryRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `AssessmentTakenQuery`.

The methods specified by the record type are available through the underlying object.

Assessment Taken Form Record

class `dlkit.assessment.records.`**`AssessmentTakenFormRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `AssessmentTakenForm`.

The methods specified by the record type are available through the underlying object.

Assessment Taken Search Record

class `dlkit.assessment.records.`**`AssessmentTakenSearchRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `AssessmentTakenSearch`.

The methods specified by the record type are available through the underlying object.

Assessment Section Record

class `dlkit.assessment.records.`**`AssessmentSectionRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `AssessmentSection`.

The methods specified by the record type are available through the underlying object.

Bank Record

class `dlkit.assessment.records.`**`BankRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `Bank`.

The methods specified by the record type are available through the underlying object.

Bank Query Record

class `dlkit.assessment.records.`**`BankQueryRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `BankQuery`.

The methods specified by the record type are available through the underlying object.

Bank Form Record

class `dlkit.assessment.records.`**`BankFormRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `BankForm`.

The methods specified by the record type are available through the underlying object.

Bank Search Record

class `dlkit.assessment.records.`**`BankSearchRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `BankSearch`.

The methods specified by the record type are available through the underlying object.

Response Record

class `dlkit.assessment.records.`**`ResponseRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `Response`.

The methods specified by the record type are available through the underlying object.

Rules

Response

class `dlkit.assessment.rules.`**`Response`**

Bases: *`dlkit.osid.rules.OsidCondition`*

A response to an assessment item.

This interface contains methods to set values in response to an assessmet item and mirrors the item record structure with the corresponding setters.

`item_id`

Gets the `Id` of the `Item`.

Returns the assessment item `Id`

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`item`

Gets the `Item`.

Returns the assessment item

Return type `osid.assessment.Item`

compliance: mandatory – This method must be implemented.

`get_response_record`(*item_record_type*)

Gets the response record corresponding to the given `Item` record `Type`.

This method is used to retrieve an object implementing the requested record. The `item_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(item_record_type)` is `true` .

Parameters **`item_record_type`** (`osid.type.Type`) – an item record type

Returns the response record

Return type `osid.assessment.records.ResponseRecord`

Raise `NullArgument` – `item_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(item_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Assessment.Authoring

Summary

Assessment Authoring Open Service Interface Definitions assessment.authoring version 3.0.0

The Assessment OSID provides the means to create, access, and take assessments. An `Assessment` may represent a quiz, survey, or other evaluation that includes assessment `Items`. The OSID defines methods to describe the flow of control and the relationships among the objects. Assessment `Items` are extensible objects to capture various types of questions, such as a multiple choice or asset submission.

The Assessment service can br broken down into several distinct services:

- Assessment Taking
- Assessment and Item authoring
- Accessing and managing banks of assessments and items

Each of these service areas are covered by different session and object interfaces. The object interfaces describe both the structure of the assessment and follow an assessment management workflow. They are:

- `Item` : a question and answer pair
- `Response:` a response to an `Item` question
- `Assessment` : a set of `Items`
- `AssessmentPart:` A grouped set of `Items` for fancier assessment sequencing
- `AssessmentOffering:` An `Assessment` available for taking
- `AssessmentTaken:` An `AssessmentOffering` that has been completed or in progress

The `AssessmentSession` is used to take an assessment and review the results. It captures various ways an assessment can be taken which may include time constraints, the ability to suspend and resume, availability of an answer key, or access to a score or other evaluation. Care should be taken to understand the various interoperability issues in using this interface.

An `AssessmentSession` may be created using an `AssessmentOffered` or `AssessmentTaken Id`. If instantiated with an `AssessmentOffered Id,` an `AssessmentTaken` is implicitly created and further references to its state should be performed using the `AssessmentTaken Id`.

An `AssessmentSession` is a mapping of an `AssessmentOffered` to an `Agent` at a point in time. The resulting `AssessmentTaken` is an identifier representing this relationship.

On the authoring side, Items map to `Assessments`. An `Item` may appear in more than one `Assessment`. Item banks may be used to catalog sets of Items and/or sets of `Assessments`. Assessment Authoring Open Service Interface Definitions assessment.authoring version 3.0.0

The Assessment OSID provides the means to create, access, and take assessments. An `Assessment` may represent a quiz, survey, or other evaluation that includes assessment `Items`. The OSID defines methods to describe the flow of control and the relationships among the objects. Assessment `Items` are extensible objects to capture various types of questions, such as a multiple choice or asset submission.

The Assessment service can br broken down into several distinct services:

- Assessment Taking
- Assessment and Item authoring
- Accessing and managing banks of assessments and items

Each of these service areas are covered by different session and object interfaces. The object interfaces describe both the structure of the assessment and follow an assessment management workflow. They are:

- `Item` : a question and answer pair
- `Response:` a response to an `Item` question
- `Assessment` : a set of `Items`
- `AssessmentPart:` A grouped set of `Items` for fancier assessment sequencing
- `AssessmentOffering:` An `Assessment` available for taking
- `AssessmentTaken:` An `AssessmentOffering` that has been completed or in progress

The `AssessmentSession` is used to take an assessment and review the results. It captures various ways an assessment can be taken which may include time constraints, the ability to suspend and resume, availability of an answer key, or access to a score or other evaluation. Care should be taken to understand the various interoperability issues in using this interface.

An `AssessmentSession` may be created using an `AssessmentOffered` or `AssessmentTaken Id`. If instantiated with an `AssessmentOffered Id`, an `AssessmentTaken` is implicitly created and further references to its state should be performed using the `AssessmentTaken Id`.

An `AssessmentSession` is a mapping of an `AssessmentOffered` to an `Agent` at a point in time. The resulting `AssessmentTaken` is an identifier representing this relationship.

On the authoring side, Items map to `Assessments`. An `Item` may appear in more than one `Assessment`. Item banks may be used to catalog sets of Items and/or sets of `Assessments`.

Service Managers

Assessment Authoring Profile

class `dlkit.services.assessment_authoring.`**`AssessmentAuthoringProfile`**

Bases: `dlkit.osid.managers.OsidProfile`

The `AssessmentAuthoringProfile` describes the interoperability among assessment authoring services.

`supports_assessment_part_lookup``()`

Tests if looking up assessment part is supported.

Returns `true` if assessment part lookup is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_assessment_part_query``()`

Tests if querying assessment part is supported.

Returns `true` if assessment part query is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_assessment_part_admin``()`

Tests if an assessment part administrative service is supported.

Returns `true` if assessment part administration is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_assessment_part_item``()`

Tests if an assessment part item service is supported for looking up assessment part and item mappings.

Returns `true` if assessment part item service is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_assessment_part_item_design``()`

Tests if an assessment part item design session is supported.

Returns `true` if an assessment part item design service is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_sequence_rule_lookup`()

Tests if looking up sequence rule is supported.

Returns `true` if sequence rule lookup is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_sequence_rule_admin`()

Tests if a sequence rule administrative service is supported.

Returns `true` if sequence rule administration is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`assessment_part_record_types`

Gets the supported `AssessmentPart` record types.

Returns a list containing the supported `AssessmentPart` record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

`assessment_part_search_record_types`

Gets the supported `AssessmentPart` search record types.

Returns a list containing the supported `AssessmentPart` search record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

`sequence_rule_record_types`

Gets the supported `SequenceRule` record types.

Returns a list containing the supported `SequenceRule` record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

`sequence_rule_search_record_types`

Gets the supported `SequenceRule` search record types.

Returns a list containing the supported `SequenceRule` search record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

`sequence_rule_enabler_record_types`

Gets the supported `SequenceRuleEnabler` record types.

Returns a list containing the supported `SequenceRuleEnabler` record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

sequence_rule_enabler_search_record_types

Gets the supported `SequenceRuleEnabler` search record types.

Returns a list containing the supported `SequenceRuleEnabler` search record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

Assessment Authoring Manager

class `dlkit.services.assessment_authoring.`**`AssessmentAuthoringManager`**(*proxy=None*)

Bases: `dlkit.osid.managers.OsidManager`, `dlkit.osid.sessions.OsidSession`, *`dlkit.services.assessment_authoring.AssessmentAuthoringProfile`*

The assessment authoring manager provides access to assessment authoring sessions and provides interoperability tests for various aspects of this service.

The sessions included in this manager are:

- `AssessmentPartLookupSession`: a session to retrieve assessment part
- `AssessmentPartQuerySession`: a session to query for assessment part
- `AssessmentPartSearchSession`: a session to search for assessment part
- `AssessmentPartAdminSession`: a session to create and delete assessment part
- `AssessmentPartNotificationSession`: a session to receive notifications pertaining to assessment part changes
- `AssessmentPartBankSession`: a session to look up assessment part bank mappings
- `AssessmentPartBankAssignmentSession`: a session to manage assessment part to bank mappings
- `AssessmentPartSmartBankSession`: a session to manage dynamic bank of assessment part
- `AssessmentPartItemSession`: a session to look up assessment part to item mappings
- `AssessmentPartItemDesignSession`: a session to map items to assessment parts
- `SequenceRuleLookupSession`: a session to retrieve sequence rule
- `SequenceRuleQuerySession`: a session to query for sequence rule
- `SequenceRuleSearchSession`: a session to search for sequence rule
- `SequenceRuleAdminSession`: a session to create and delete sequence rule
- `SequenceRuleNotificationSession`: a session to receive notifications pertaining to sequence rule changes
- `SequenceRuleBankSession`: a session to look up sequence rule bank mappings
- `SequenceRuleBankAssignmentSession`: a session to manage sequence rule to bank mappings
- `SequenceRuleSmartBankSession`: a session to manage dynamic bank of sequence rule
- `SequenceRuleEnablerLookupSession`: a session to retrieve sequence rule enablers
- `SequenceRuleEnablerQuerySession`: a session to query for sequence rule enablers
- `SequenceRuleEnablerSearchSession`: a session to search for sequence rule enablers
- `SequenceRuleEnablerAdminSession`: a session to create and delete sequence rule enablers

- `SequenceRuleEnablerNotificationSession`: a session to receive notifications pertaining to sequence rule enabler changes
- `SequenceRuleEnablerBankSession`: a session to look up sequence rule enabler bank mappings
- `SequenceRuleEnablerBankAssignmentSession`: a session to manage sequence rule enabler to bank mappings
- `SequenceRuleEnablerSmartBankSession`: a session to manage dynamic bank of sequence rule enablers
- `SequenceRuleEnableRuleLookupSession`: a session to look up sequence rule enabler mappings
- `SequenceRuleEnablerRuleApplicationSession`: a session to apply sequence rule enablers

Objects

Assessment Part

class `dlkit.assessment_authoring.objects.`**`AssessmentPart`**

Bases: *`dlkit.osid.objects.OsidObject`*, *`dlkit.osid.markers.Containable`*, *`dlkit.osid.markers.Operable`*

An `AssessmentPart` represents a section of an assessment.

`AssessmentParts` may be visible as sections of an assessment or just used to clump together a set of items on which to hang sequence rules.

`assessment_id`

Gets the assessment `Id` to which this rule belongs.

Returns `Id` of an assessment

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`assessment`

Gets the assessment to which this rule belongs.

Returns an assessment

Return type `osid.assessment.Assessment`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`has_parent_part``()`

Tests if this assessment part belongs to a parent assessment part.

Returns `true` if this part has a parent, `false` if a root

Return type `boolean`

compliance: mandatory – This method must be implemented.

`assessment_part_id`

Gets the parent assessment `Id`.

Returns `Id` of an assessment

Return type `osid.id.Id`

Raise `IllegalState` – `has_parent_part()` is `false`

compliance: mandatory – This method must be implemented.

`assessment_part`

Gets the parent assessment.

Returns the parent assessment part

Return type `osid.assessment.authoring.AssessmentPart`

Raise `IllegalState` – `has_parent_part()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`is_section``()`

Tests if this part should be visible as a section in an assessment.

If visible, this part will appear to the user as a separate section of the assessment. Typically, a section may not be under a non-sectioned part.

Returns `true` if this part is a section, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`weight`

Gets an integral weight factor for this assessment part used for scoring.

The percentage weight for this part is this weight divided by the sum total of all the weights in the assessment.

Returns the weight

Return type `cardinal`

compliance: mandatory – This method must be implemented.

`allocated_time`

Gets the allocated time for this part.

The allocated time may be used to assign fixed time limits to each item or can be used to estimate the total assessment time.

Returns allocated time

Return type `osid.calendaring.Duration`

compliance: mandatory – This method must be implemented.

`child_assessment_part_ids`

Gets any child assessment part `Ids`.

Returns `Ids` of the child assessment parts

Return type `osid.id.IdList`

compliance: mandatory – This method must be implemented.

`child_assessment_parts`

Gets any child assessment parts.

Returns the child assessment parts

Return type `osid.assessment.authoring.AssessmentPartList`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

get_assessment_part_record(*assessment_part_record_type*)
Gets the assessment part record corresponding to the given `AssessmentPart` record `Type`.

This method is used to retrieve an object implementing the requested record. The `assessment_part_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(assessment_part_record_type)` is `true` .

Parameters **assessment_part_record_type** (`osid.type.Type`) – the type of the record to retrieve

Returns the assessment part record

Return type `osid.assessment.authoring.records.AssessmentPartRecord`

Raise `NullArgument` – `assessment_part_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(assessment_part_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Assessment Part Form

class `dlkit.assessment_authoring.objects.`**AssessmentPartForm**
Bases: *dlkit.osid.objects.OsidObjectForm*, *dlkit.osid.objects.OsidContainableForm*, *dlkit.osid.objects.OsidOperableForm*

This is the form for creating and updating `AssessmentParts`.

Like all `OsidForm` objects, various data elements may be set here for use in the create and update methods in the `AssessmentAuthoringSession`. For each data element that may be set, metadata may be examined to provide display hints or data constraints.

weight_metadata
Gets the metadata for the weight.

Returns metadata for the weight

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

weight

allocated_time_metadata
Gets the metadata for the allocated time.

Returns metadata for the allocated time

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

allocated_time

get_assessment_part_form_record(*assessment_part_record_type*)
Gets the `AssessmentPartFormRecord` corresponding to the given assessment record `Type`.

Parameters **`assessment_part_record_type`** (`osid.type.Type`) – the assessment part record type

Returns the assessment part record

Return type `osid.assessment.authoring.records.AssessmentPartFormRecord`

Raise `NullArgument` – `assessment_part_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(assessment_part_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Assessment Part List

class `dlkit.assessment_authoring.objects.`**`AssessmentPartList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists, AssessmentPartList` provides a means for accessing `AssessmentPart` elements sequentially either one at a time or many at a time.

Examples: while (apl.hasNext()) { AssessmentPart assessmentPart = apl.getNextAssessmentPart(); }

or

while (apl.hasNext()) { AssessmentPart[] assessmentParts = apl.hetNextAssessmentParts(apl.available());

}

`next_assessment_part`

Gets the next `AssessmentPart` in this list.

Returns the next `AssessmentPart` in this list. The `has_next()` method should be used to test that a next `AssessmentPart` is available before calling this method.

Return type `osid.assessment.authoring.AssessmentPart`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_assessment_parts`(*n*)

Gets the next set of `AssessmentPart` elements in this list which must be less than or equal to the number returned from `available()`.

Parameters **`n`** (`cardinal`) – the number of `AssessmentPart` elements requested which should be less than or equal to `available()`

Returns an array of `AssessmentPart` elements.The length of the array is less than or equal to the number specified.

Return type `osid.assessment.authoring.AssessmentPart`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Sequence Rule

class `dlkit.assessment_authoring.objects.`**`SequenceRule`**

Bases: *`dlkit.osid.objects.OsidRule`*

A `SequenceRule` defines the ordering of `AssessmentParts`.

`assessment_part_id`

Gets the assessment part `Id` to which this rule belongs.

Returns `Id` of an assessment part

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`assessment_part`

Gets the assessment part to which this rule belongs.

Returns an assessment part

Return type `osid.assessment.authoring.AssessmentPart`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`next_assessment_part_id`

Gets the next assessment part `Id` for success of this rule.

Returns `Id` of an assessment part

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`next_assessment_part`

Gets the next assessment part for success of this rule.

Returns an assessment part

Return type `osid.assessment.authoring.AssessmentPart`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`minimum_score`

Gets the minimum score expressed as an integer (0-100) for this rule.

Returns minimum score

Return type `cardinal`

compliance: mandatory – This method must be implemented.

`maximum_score`

Gets the maximum score expressed as an integer (0-100) for this rule.

Returns maximum score

Return type `cardinal`

compliance: mandatory – This method must be implemented.

`is_cumulative``()`

Tests if the score is applied to all previous assessment parts.

Returns `true` if the score is applied to all previous assessment parts, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`applied_assessment_part_ids`

Qualifies `is_cumulative()` to apply to a specific list of assessment parts.

If `is_cumulative()` is `true,` this method may return an empty list to mean all previous assessment parts.

Returns list of assessment parts

Return type `osid.id.IdList`

Raise `IllegalState` – `is_cumulative()` is `false`

compliance: mandatory – This method must be implemented.

`applied_assessment_parts`

Qualifies `is_cumulative()` to apply to a specific list of assessment parts.

If `is_cumulative()` is `true,` this method may return an empty list to mean all previous assessment parts.

Returns list of assessment parts

Return type `osid.assessment.authoring.AssessmentPartList`

Raise `IllegalState` – `is_cumulative()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_sequence_rule_record`(*sequence_rule_record_type*)

Gets the assessment sequence rule record corresponding to the given `SequenceRule` record `Type`.

This method is used to retrieve an object implementing the requested record. The `sequence_rule_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(sequence_rule_record_type)` is `true` .

Parameters **`sequence_rule_record_type`** (`osid.type.Type`) – the type of the record to retrieve

Returns the assessment sequence rule record

Return type `osid.assessment.authoring.records.SequenceRuleRecord`

Raise `NullArgument` – `sequence_rule_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(sequence_rule_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Sequence Rule Form

class `dlkit.assessment_authoring.objects.`**`SequenceRuleForm`**

Bases: *`dlkit.osid.objects.OsidRuleForm`*

This is the form for creating and updating sequence rules.

Like all `OsidForm` objects, various data elements may be set here for use in the create and update methods in the `SequenceSession` For each data element that may be set, metadata may be examined to provide display hints or data constraints.

`minimum_score_metadata`
Gets the metadata for the minimum score.

Returns metadata for the minimum score

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`minimum_score`

`maximum_score_metadata`
Gets the metadata for the maximum score.

Returns metadata for the maximum score

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`maximum_score`

`cumulative_metadata`
Gets the metadata for the cumulative flag.

Returns metadata for the cumulative flag

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`cumulative`

`applied_assessment_parts_metadata`
Gets the metadata for the applied assessment parts.

Returns metadata for the applied assessment parts

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`apply_assessment_parts`(*assessment_part_ids*)
Designates assessment parts to which the rule applies.

Parameters **`assessment_part_ids`** (`osid.id.Id[]`) – the parts to which this rule should apply

Raise `InvalidArgument` – `assessment_part_ids` is invalid

Raise `NoAccess` – `Metadata.isReadOnly()` is `true`

Raise `NullArgument` – `assessment_part_ids` is `null`

compliance: mandatory – This method must be implemented.

`get_sequence_rule_form_record`(*sequence_rule_record*)
Gets the `SequenceRuleFormRecord` corresponding to the given sequence rule record `Type`.

Parameters **`sequence_rule_record`** (`osid.type.Type`) – a sequence rule record type

Returns the sequence rule record

Return type `osid.assessment.authoring.records.SequenceRuleFormRecord`

Raise `NullArgument` – `sequence_rule_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(asequence_rule_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Sequence Rule List

class `dlkit.assessment_authoring.objects.`**`SequenceRuleList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists`, `SequenceRuleList` provides a means for accessing `SequenceRule` elements sequentially either one at a time or many at a time.

Examples: while (srl.hasNext()) { AssessmentSequenceRule rule = srl.getNextAssessmentSequenceRule(); }

or

while (srl.hasNext()) { AssessmentSequenceRule[] rules = srl.getNextAssessmentSequenceRules(srl.available());

}

`next_sequence_rule`

Gets the next `SequenceRule` in this list.

Returns the next `SequenceRule` in this list. The `has_next()` method should be used to test that a next `SequenceRule` is available before calling this method.

Return type `osid.assessment.authoring.SequenceRule`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_sequence_rules`(*n*)

Gets the next set of `SequenceRule` elements in this list which must be less than or equal to the number returned from `available()`.

Parameters **n** (`cardinal`) – the number of `SequenceRule` elements requested which should be less than or equal to `available()`

Returns an array of `SequenceRule` elements.The length of the array is less than or equal to the number specified.

Return type `osid.assessment.authoring.SequenceRule`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Queries

Assessment Part Query

class `dlkit.assessment_authoring.queries.`**`AssessmentPartQuery`**

Bases: *`dlkit.osid.queries.OsidObjectQuery`*, *`dlkit.osid.queries.OsidContainableQuery`*, *`dlkit.osid.queries.OsidOperableQuery`*

This is the query for searching assessment parts.

Each method match request produces an `AND` term while multiple invocations of a method produces a nested `OR`.

`match_assessment_id`(*assessment_id*, *match*)

Sets the assessment `Id` for this query.

Parameters

- **`assessment_id`** (`osid.id.Id`) – an assessment `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `assessment_id` is `null`

compliance: mandatory – This method must be implemented.

`assessment_id_terms`

`supports_assessment_query`()

Tests if an `AssessmentQuery` is available.

Returns `true` if an assessment query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`assessment_query`

Gets the query for an assessment.

Multiple retrievals produce a nested `OR` term.

Returns the assessment query

Return type `osid.assessment.AssessmentQuery`

Raise `Unimplemented` – `supports_assessment_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_assessment_query()‘‘ is ‘‘true‘‘.

`assessment_terms`

`match_parent_assessment_part_id`(*assessment_part_id*, *match*)

Sets the assessment part `Id` for this query.

Parameters

- **`assessment_part_id`** (`osid.id.Id`) – an assessment part `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `assessment_part_id` is `null`

compliance: mandatory – This method must be implemented.

`parent_assessment_part_id_terms`

`supports_parent_assessment_part_query`()
Tests if an `AssessmentPartQuery` is available.

Returns `true` if an assessment part query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`parent_assessment_part_query`
Gets the query for an assessment part.

Multiple retrievals produce a nested `OR` term.

Returns the assessment part query

Return type `osid.assessment.authoring.AssessmentPartQuery`

Raise `Unimplemented` – `supports_parent_assessment_part_query()` is `false`

compliance: optional – This method must be implemented if ``supports_parent_assessment_part_query()`` is ``true``.

`match_any_parent_assessment_part`(*match*)
Matches assessment parts with any parent assessment part.

Parameters **`match`** (`boolean`) – `true` to match assessment parts with any parent, `false` to match assessment parts with no parents

compliance: mandatory – This method must be implemented.

`parent_assessment_part_terms`

`match_section`(*match*)
Matches assessment parts that are also used as sections.

Parameters **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

compliance: mandatory – This method must be implemented.

`section_terms`

`match_weight`(*low*, *high*, *match*)
Matches assessment parts that fall in between the given weights inclusive.

Parameters

- **`low`** (`cardinal`) – low end of range
- **`high`** (`cardinal`) – high end of range
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `high` is less than `low`

compliance: mandatory – This method must be implemented.

`match_any_weight`(*match*)
Matches assessment parts with any weight assigned.

Parameters **`match`** (`boolean`) – `true` to match assessment parts with any wieght, `false` to match assessment parts with no weight

compliance: mandatory – This method must be implemented.

`weight_terms`

match_allocated_time(*low*, *high*, *match*)
Matches assessment parts hose allocated time falls in between the given times inclusive.

Parameters

- **low** (osid.calendaring.Duration) – low end of range
- **high** (osid.calendaring.Duration) – high end of range
- **match** (boolean) – true for a positive match, false for a negative match

Raise InvalidArgument – high is less than low

compliance: mandatory – This method must be implemented.

match_any_allocated_time(*match*)
Matches assessment parts with any time assigned.

Parameters **match** (boolean) – true to match assessment parts with any alloocated time, false to match assessment parts with no allocated time

compliance: mandatory – This method must be implemented.

allocated_time_terms

match_child_assessment_part_id(*assessment_part_id*, *match*)
Sets the assessment part Id for this query.

Parameters

- **assessment_part_id** (osid.id.Id) – an assessment part Id
- **match** (boolean) – true for a positive match, false for a negative match

Raise NullArgument – assessment_part_id is null

compliance: mandatory – This method must be implemented.

child_assessment_part_id_terms

supports_child_assessment_part_query()
Tests if an AssessmentPartQuery is available.

Returns true if an assessment part query is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

child_assessment_part_query
Gets the query for an assessment part.

Multiple retrievals produce a nested OR term.

Returns the assessment part query

Return type osid.assessment.authoring.AssessmentPartQuery

Raise Unimplemented – supports_child_assessment_part_query() is false

compliance: optional – This method must be implemented if ``supports_child_assessment_part_query()`` is ``true``.

match_any_child_assessment_part(*match*)
Matches assessment parts with any child assessment part.

Parameters **match** (boolean) – true to match assessment parts with any children, false to match assessment parts with no children

compliance: mandatory – This method must be implemented.

`child_assessment_part_terms`

`match_bank_id`(*bank_id*, *match*)
Matches constrainers mapped to the bank.

Parameters

- **`bank_id`** (`osid.id.Id`) – the bank `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `bank_id` is `null`

compliance: mandatory – This method must be implemented.

`bank_id_terms`

`supports_bank_query`()
Tests if an `BankQuery` is available.

Returns `true` if a bank query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`bank_query`
Gets the query for a bank.

Multiple retrievals produce a nested `OR` term.

Returns the bank query

Return type `osid.assessment.BankQuery`

Raise `Unimplemented` – `supports_bank_query()` is `false`

compliance: optional – This method must be implemented if ``supports_bank_query()`` is ``true``.

`bank_terms`

`get_assessment_part_query_record`(*assessment_part_record_type*)
Gets the assessment part query record corresponding to the given `AssessmentPart` record `Type`.

Multiple retrievals produce a nested `OR` term.

Parameters **`assessment_part_record_type`** (`osid.type.Type`) – an assessment part record type

Returns the assessment part query record

Return type `osid.assessment.authoring.records.AssessmentPartQueryRecord`

Raise `NullArgument` – `assessment_part_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(assessment_part_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Sequence Rule Query

class `dlkit.assessment_authoring.queries.`**`SequenceRuleQuery`**

Bases: *`dlkit.osid.queries.OsidRuleQuery`*

This is the query for searching sequence rules.

Each method match specifies a `AND` term while multiple invocations of the same method produce a nested `OR`.

`match_assessment_part_id`(*assessment_part_id*, *match*)

Sets the assessment part `Id` for this query.

Parameters

- **`assessment_part_id`** (`osid.id.Id`) – an assessment part `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `assessment_part_id` is `null`

compliance: mandatory – This method must be implemented.

`assessment_part_id_terms`

`supports_assessment_part_query`()

Tests if an `AssessmentPartQuery` is available.

Returns `true` if an assessment part query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`assessment_part_query`

Gets the query for an assessment part.

Multiple retrievals produce a nested `OR` term.

Returns the assessment part query

Return type `osid.assessment.authoring.AssessmentPartQuery`

Raise `Unimplemented` – `supports_assessment_part_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_assessment_part_query()‘‘ is ‘‘true‘‘.

`assessment_part_terms`

`match_next_assessment_part_id`(*assessment_part_id*, *match*)

Sets the assessment part `Id` for this query.

Parameters

- **`assessment_part_id`** (`osid.id.Id`) – an assessment part `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `assessment_part_id` is `null`

compliance: mandatory – This method must be implemented.

`next_assessment_part_id_terms`

`supports_next_assessment_part_query`()

Tests if an `AssessmentPartQuery` is available.

Returns `true` if an assessment part query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`next_assessment_part_query`

Gets the query for an assessment part.

Multiple retrievals produce a nested `OR` term.

Returns the assessment part query

Return type `osid.assessment.authoring.AssessmentPartQuery`

Raise `Unimplemented` – `supports_next_assessment_part_query()` is `false`

compliance: optional – This method must be implemented if ``supports_next_assessment_part_query()`` is ``true``.

`next_assessment_part_terms`

`match_minimum_score`(*low*, *high*, *match*)

Matches minimum scores that fall in between the given scores inclusive.

Parameters

- **`low`** (`cardinal`) – low end of range
- **`high`** (`cardinal`) – high end of range
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `high` is less than `low`

compliance: mandatory – This method must be implemented.

`match_any_minimum_score`(*match*)

Matches assessment parts with any minimum score assigned.

Parameters **`match`** (`boolean`) – `true` to match assessment parts with any minimum score, `false` to match assessment parts with no minimum score

compliance: mandatory – This method must be implemented.

`minimum_score_terms`

`match_maximum_score`(*low*, *high*, *match*)

Matches maximum scores that fall in between the given scores inclusive.

Parameters

- **`low`** (`cardinal`) – low end of range
- **`high`** (`cardinal`) – high end of range
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `high` is less than `low`

compliance: mandatory – This method must be implemented.

`match_any_maximum_score`(*match*)

Matches assessment parts with any maximum score assigned.

Parameters **`match`** (`boolean`) – `true` to match assessment parts with any maximum score, `false` to match assessment parts with no maximum score

compliance: mandatory – This method must be implemented.

`maximum_score_terms`

match_cumulative(*match*)

Matches cumulative rules.

Parameters **match** (boolean) – true for a positive match, false for a negative match

compliance: mandatory – This method must be implemented.

cumulative_terms

match_applied_assessment_part_id(*assessment_part_id*, *match*)

Sets the assessment part Id for this query.

Parameters

- **assessment_part_id** (osid.id.Id) – an assessment part Id
- **match** (boolean) – true for a positive match, false for a negative match

Raise NullArgument – assessment_part_id is null

compliance: mandatory – This method must be implemented.

applied_assessment_part_id_terms

supports_applied_assessment_part_query()

Tests if an AssessmentPartQuery is available.

Returns true if an assessment part query is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

applied_assessment_part_query

Gets the query for an assessment part.

Multiple retrievals produce a nested OR term.

Returns the assessment part query

Return type osid.assessment.authoring.AssessmentPartQuery

Raise Unimplemented – supports_applied_assessment_part_query() is false

compliance: optional – This method must be implemented if ``supports_applied_assessment_part_query()`` is ``true``.

match_any_applied_assessment_part(*match*)

Matches assessment parts with any applied assessment part.

Parameters **match** (boolean) – true to match assessment parts with any applied assessment part, false to match assessment parts with no applied assessment parts

compliance: mandatory – This method must be implemented.

applied_assessment_part_terms

match_bank_id(*bank_id*, *match*)

Matches constrainers mapped to the bank.

Parameters

- **bank_id** (osid.id.Id) – the bank Id
- **match** (boolean) – true for a positive match, false for a negative match

Raise NullArgument – bank_id is null

compliance: mandatory – This method must be implemented.

bank_id_terms

supports_bank_query()

Tests if an `BankQuery` is available.

Returns `true` if a bank query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

bank_query

Gets the query for a bank.

Multiple retrievals produce a nested `OR` term.

Returns the bank query

Return type `osid.assessment.BankQuery`

Raise `Unimplemented` – `supports_bank_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_bank_query()‘‘ is ‘‘true‘‘.

bank_terms

get_sequence_rule_query_record(*sequence_rule_record_type*)

Gets the sequence rule query record corresponding to the given `SequenceRule` record `Type`.

Multiple record retrievals produce a nested `OR` term.

Parameters **sequence_rule_record_type** (`osid.type.Type`) – a sequence rule record type

Returns the sequence rule query record

Return type `osid.assessment.authoring.records.SequenceRuleQueryRecord`

Raise `NullArgument` – `sequence_rule_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(sequence_rule_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Records

Assessment Part Record

class `dlkit.assessment_authoring.records.`**AssessmentPartRecord**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `AssessmentPart`.

The methods specified by the record type are available through the underlying object.

Assessment Part Query Record

class `dlkit.assessment_authoring.records.`**`AssessmentPartQueryRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `AssessmentPartQuery`.

The methods specified by the record type are available through the underlying object.

Assessment Part Form Record

class `dlkit.assessment_authoring.records.`**`AssessmentPartFormRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `AssessmentPartForm`.

The methods specified by the record type are available through the underlying object.

Assessment Part Search Record

class `dlkit.assessment_authoring.records.`**`AssessmentPartSearchRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `AssessmentPartSearch`.

The methods specified by the record type are available through the underlying object.

Sequence Rule Record

class `dlkit.assessment_authoring.records.`**`SequenceRuleRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `SequenceRule`.

The methods specified by the record type are available through the underlying object.

Sequence Rule Query Record

class `dlkit.assessment_authoring.records.`**`SequenceRuleQueryRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `SequenceRuleQuery`.

The methods specified by the record type are available through the underlying object.

Sequence Rule Form Record

class `dlkit.assessment_authoring.records.`**`SequenceRuleFormRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `SequenceRuleForm`.

The methods specified by the record type are available through the underlying object.

Sequence Rule Search Record

class `dlkit.assessment_authoring.records.`**`SequenceRuleSearchRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `SequenceRuleSearch`.

The methods specified by the record type are available through the underlying object.

Commenting

Summary

Commenting Open Service Interface Definitions commenting version 3.0.0

The Commenting OSID provides a means of relating user comments and ratings to OSID Objects.

The Commenting OSID may be used as an auxiliary service orchestrated with other OSIDs to either provide administrative comments as well as create a social network-esque comment and rating service to various `OsidObjects`.

Comments

`Comments` contain text entries logged by date and `Agent`. A `Comment` may also include a rating represented by a `Grade` defined in a `GradeSystem`. The `RatingLookupSession` may be used to query cumulative scores across an object reference or the entire `Book`.

`Comments` are `OsidRelationships` between a commentor and a reference Id. The relationship defines dates for which the comment and/or rating is effective.

Commentors

An `Agent` comments on something. As a person is represented by a `Resource` in the Resource OSID, the Comments provide access to both the commenting `Agent` and the related `Resource` to avoid the need of an additional service orchestration for resolving the `Agent`.

Cataloging

`Comments` are cataloged in `Books` which may also be grouped hierarchically to federate multiple collections of comments.

Sub Packages

The Commenting OSID includes a Commenting Batch OSID for managing `Comments` and `Books` in bulk. Commenting Open Service Interface Definitions commenting version 3.0.0

The Commenting OSID provides a means of relating user comments and ratings to OSID Objects.

The Commenting OSID may be used as an auxiliary service orchestrated with other OSIDs to either provide administrative comments as well as create a social network-esque comment and rating service to various `OsidObjects`.

Comments

`Comments` contain text entries logged by date and `Agent`. A `Comment` may also include a rating represented by a `Grade` defined in a `GradeSystem`. The `RatingLookupSession` may be used to query cumulative scores across an object reference or the entire `Book`.

`Comments` are `OsidRelationships` between a commentor and a reference Id. The relationship defines dates for which the comment and/or rating is effective.

Commentors

An `Agent` comments on something. As a person is represented by a `Resource` in the Resource OSID, the Comments provide access to both the commenting `Agent` and the related `Resource` to avoid the need of an additional service orchestration for resolving the `Agent`.

Cataloging

`Comments` are cataloged in `Books` which may also be grouped hierarchically to federate multiple collections of comments.

Sub Packages

The Commenting OSID includes a Commenting Batch OSID for managing `Comments` and `Books` in bulk.

Service Managers

Commenting Profile

class `dlkit.services.commenting.`**`CommentingProfile`**

Bases: `dlkit.osid.managers.OsidProfile`

The commenting profile describes the interoperability among commenting services.

`supports_comment_lookup``()`

Tests for the availability of a comment lookup service.

Returns `true` if comment lookup is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_comment_query``()`

Tests if querying comments is available.

Returns `true` if comment query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_comment_admin``()`

Tests if managing comments is available.

Returns `true` if comment admin is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_book_lookup``()`

Tests for the availability of an book lookup service.

Returns `true` if book lookup is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_book_admin``()`

Tests for the availability of a book administrative service for creating and deleting books.

Returns `true` if book administration is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_book_hierarchy()
Tests for the availability of a book hierarchy traversal service.

Returns `true` if book hierarchy traversal is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_book_hierarchy_design()
Tests for the availability of a book hierarchy design service.

Returns `true` if book hierarchy design is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented in all providers.

comment_record_types
Gets the supported `Comment` record types.

Returns a list containing the supported comment record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

comment_search_record_types
Gets the supported comment search record types.

Returns a list containing the supported comment search record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

book_record_types
Gets the supported `Book` record types.

Returns a list containing the supported book record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

book_search_record_types
Gets the supported book search record types.

Returns a list containing the supported book search record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

Commenting Manager

class dlkit.services.commenting.**CommentingManager**(*proxy=None*)
Bases: `dlkit.osid.managers.OsidManager`, `dlkit.osid.sessions.OsidSession`, *dlkit.services.commenting.CommentingProfile*

The commenting manager provides access to commenting sessions and provides interoperability tests for various aspects of this service.

The sessions included in this manager are:

- `CommentLookupSession`: a session to lookup comments
- `RatingLookupSession`: a session to lookup comments
- `CommentQuerySession`: a session to query comments
- `CommentSearchSession`: a session to search comments
- `CommentAdminSession`: a session to manage comments
- `CommentNotificationSession`: a session to subscribe to notifications of comment changes
- `CommentBookSession`: a session for looking up comment and book mappings
- `CommentBookAssignmentSession`: a session for managing comment and book mappings
- `CommentSmartBookSession`: a session to manage dynamic comment books
- `BookLookupSession`: a session to retrieve books
- `BookQuerySession`: a session to query books
- `BookSearchSession`: a session to search for books
- `BookAdminSession`: a session to create, update and delete books
- `BookNotificationSession`: a session to receive notifications for changes in books
- `BookHierarchyTraversalSession`: a session to traverse hierarchies of books
- `BookHierarchyDesignSession`: a session to manage hierarchies of books

The commenting manager also provides a profile for determing the supported search types supported by this service.

`commenting_batch_manager`

Gets a `CommentingBatchManager`.

Returns a `CommentingBatchManager`

Return type `osid.commenting.batch.CommentingBatchManager`

Raise `OperationFailed` – unable to complete request

Raise `Unimplemented` – `supports_commenting_batch()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_commenting_batch()‘‘ is ‘‘true‘‘.

Book Lookup Methods

`CommentingManager.`**`can_lookup_books`**`()`

Tests if this user can perform `Book` lookups.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may not offer lookup operations to unauthorized users.

Returns `false` if lookup methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`use_comparative_book_view`**`()`

The returns from the book methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`CommentingManager.`**`use_plenary_book_view`**`()`

A complete view of the `Book` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`CommentingManager.`**`get_book`**`(`*book_id*`)`

Gets the `Book` specified by its `Id`.

In plenary mode, the exact `Id` is found or a `NotFound` results. Otherwise, the returned `Book` may have a different `Id` than requested, such as the case where a duplicate `Id` was assigned to a `Book` and retained for compatibility.

Parameters **`book_id`** (`osid.id.Id`) – `Id` of the `Book`

Returns the book

Return type `osid.commenting.Book`

Raise `NotFound` – `book_id` not found

Raise `NullArgument` – `book_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method is must be implemented.

`CommentingManager.`**`get_books_by_ids`**`(`*book_ids*`)`

Gets a `BookList` corresponding to the given `IdList`.

In plenary mode, the returned list contains all of the books specified in the `Id` list, in the order of the list, including duplicates, or an error results if an `Id` in the supplied list is not found or inaccessible. Otherwise, inaccessible `Books` may be omitted from the list and may present the elements in any order including returning a unique set.

Parameters **`book_ids`** (`osid.id.IdList`) – the list of `Ids` to retrieve

Returns the returned `Book` list

Return type `osid.commenting.BookList`

Raise `NotFound` – an `Id was` not found

Raise `NullArgument` – `book_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`get_books_by_genus_type`**`(`*book_genus_type*`)`

Gets a `BookList` corresponding to the given book genus `Type` which does not include books of genus types derived from the specified `Type`.

In plenary mode, the returned list contains all known books or an error results. Otherwise, the returned list may contain only those books that are accessible through this session.

Parameters **`book_genus_type`** (`osid.type.Type`) – a book genus type

Returns the returned `Book` list

Return type `osid.commenting.BookList`

Raise `NullArgument` – `book_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`get_books_by_parent_genus_type`**(*book_genus_type*)

Gets a `BookList` corresponding to the given book genus `Type` and include any additional books with genus types derived from the specified `Type`.

In plenary mode, the returned list contains all known books or an error results. Otherwise, the returned list may contain only those books that are accessible through this session.

Parameters **`book_genus_type`** (`osid.type.Type`) – a book genus type

Returns the returned `Book` list

Return type `osid.commenting.BookList`

Raise `NullArgument` – `book_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`get_books_by_record_type`**(*book_record_type*)

Gets a `BookList` containing the given book record `Type`.

In plenary mode, the returned list contains all known books or an error results. Otherwise, the returned list may contain only those books that are accessible through this session.

Parameters **`book_record_type`** (`osid.type.Type`) – a book record type

Returns the returned `Book` list

Return type `osid.commenting.BookList`

Raise `NullArgument` – `book_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`get_books_by_provider`**(*resource_id*)

Gets a `BookList` from the given provider ‘‘‘‘.

In plenary mode, the returned list contains all known books or an error results. Otherwise, the returned list may contain only those books that are accessible through this session.

Parameters **`resource_id`** (`osid.id.Id`) – a resource `Id`

Returns the returned `Book` list

Return type `osid.commenting.BookList`

Raise `NullArgument` – `resource_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`books`**

Gets all `Books`.

In plenary mode, the returned list contains all known books or an error results. Otherwise, the returned list may contain only those books that are accessible through this session.

Returns a list of `Books`

Return type `osid.commenting.BookList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Book Admin Methods

`CommentingManager.`**`can_create_books`**`()`

Tests if this user can create `Books`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known creating a `Book` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer create operations to unauthorized users.

Returns `false` if `Book` creation is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`can_create_book_with_record_types`**`(`*book_record_types*`)`

Tests if this user can create a single `Book` using the desired record types.

While `CommentingManager.getBookRecordTypes()` can be used to examine which records are supported, this method tests which record(s) are required for creating a specific `Book`. Providing an empty array tests if a `Book` can be created with no records.

Parameters **`book_record_types`** (`osid.type.Type[]`) – array of book record types

Returns `true` if `Book` creation using the specified record `Types` is supported, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `book_record_types` is `null`

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`get_book_form_for_create`**`(`*book_record_types*`)`

Gets the book form for creating new books.

A new form should be requested for each create transaction.

Parameters **`book_record_types`** (`osid.type.Type[]`) – array of book record types

Returns the book form

Return type `osid.commenting.BookForm`

Raise `NullArgument` – `book_record_types` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – unable to get form for requested record types

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`create_book`**`(`*book_form*`)`

Creates a new `Book`.

Parameters **`book_form`** (`osid.commenting.BookForm`) – the form for this `Book`

Returns the new `Book`

Return type `osid.commenting.Book`

Raise `IllegalState` – `book_form` already used in a create transaction

Raise `InvalidArgument` – one or more of the form elements is invalid

Raise `NullArgument` – `book_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `book_form` did not originte from `get_book_form_for_create()`

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`can_update_books`**`()`

Tests if this user can update `Books`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating a `Book` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer update operations to unauthorized users.

Returns `false` if `Book` modification is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`get_book_form_for_update`**`(`*book_id*`)`

Gets the book form for updating an existing book.

A new book form should be requested for each update transaction.

Parameters **`book_id`** (`osid.id.Id`) – the `Id` of the `Book`

Returns the book form

Return type `osid.commenting.BookForm`

Raise `NotFound` – `book_id` is not found

Raise `NullArgument` – `book_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`update_book`**`(`*book_form*`)`

Updates an existing book.

Parameters **`book_form`** (`osid.commenting.BookForm`) – the form containing the elements to be updated

Raise `IllegalState` – `book_form` already used in an update transaction

Raise `InvalidArgument` – the form contains an invalid value

Raise `NullArgument` – `book_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `book_form` did not originte from `get_book_form_for_update()`

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`can_delete_books`**`()`

Tests if this user can delete `Books` A return of true does not guarantee successful authorization.

A return of false indicates that it is known deleting a `Book` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer delete operations to unauthorized users.

Returns `false` if `Book` deletion is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`delete_book`**`(`*book_id*`)`

Deletes a `Book`.

Parameters **`book_id`** (`osid.id.Id`) – the `Id` of the `Book` to remove

Raise `NotFound` – `book_id` not found

Raise `NullArgument` – `book_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`can_manage_book_aliases`**`()`

Tests if this user can manage `Id` aliases for `Books`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known changing an alias will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer alias operations to an unauthorized user.

Returns `false` if `Book` aliasing is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`alias_book`**`(`*book_id*, *alias_id*`)`

Adds an `Id` to a `Book` for the purpose of creating compatibility.

The primary `Id` of the `Book` is determined by the provider. The new `Id` performs as an alias to the primary `Id`. If the alias is a pointer to another book, it is reassigned to the given book `Id`.

Parameters

- **book_id** (`osid.id.Id`) – the `Id` of a `Book`
- **alias_id** (`osid.id.Id`) – the alias `Id`

Raise `AlreadyExists` – `alias_id` is already assigned

Raise `NotFound` – `book_id` not found

Raise `NullArgument` – `book_id` or `alias_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Book Hierarchy Methods

`CommentingManager.`**`book_hierarchy_id`**

Gets the hierarchy `Id` associated with this session.

Returns the hierarchy `Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`book_hierarchy`**

Gets the hierarchy associated with this session.

Returns the hierarchy associated with this session

Return type `osid.hierarchy.Hierarchy`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`can_access_book_hierarchy`**`()`

Tests if this user can perform hierarchy queries.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations.

Returns `false` if hierarchy traversal methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`use_comparative_book_view`**`()`

The returns from the book methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`CommentingManager.`**`use_plenary_book_view`**`()`

A complete view of the `Book` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`CommentingManager.`**`root_book_ids`**

Gets the root book `Ids` in this hierarchy.

Returns the root book `Ids`

Return type `osid.id.IdList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`root_books`**

Gets the root books in the book hierarchy.

A node with no parents is an orphan. While all book `Ids` are known to the hierarchy, an orphan does not appear in the hierarchy unless explicitly added as a root node or child of another node.

Returns the root books

Return type `osid.commenting.BookList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method is must be implemented.

`CommentingManager.`**`has_parent_books`**`(`*book_id*`)`

Tests if the `Book` has any parents.

Parameters **`book_id`** (`osid.id.Id`) – a book `Id`

Returns `true` if the book has parents, f `alse` otherwise

Return type `boolean`

Raise `NotFound` – `book_id` is not found

Raise `NullArgument` – `book_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`is_parent_of_book`**`(`*id_*, *book_id*`)`

Tests if an `Id` is a direct parent of book.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`book_id`** (`osid.id.Id`) – the `Id` of a book

Returns `true` if this `id` is a parent of `book_id,` f `alse` otherwise

Return type `boolean`

Raise `NotFound` – `book_id` is not found

Raise `NullArgument` – `id` or `book_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` not found return `false`.

`CommentingManager.`**`get_parent_book_ids`**`(`*book_id*`)`
Gets the parent `Ids` of the given book.

Parameters **`book_id`** (`osid.id.Id`) – a book `Id`

Returns the parent `Ids` of the book

Return type `osid.id.IdList`

Raise `NotFound` – `book_id` is not found

Raise `NullArgument` – `book_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`get_parent_books`**`(`*book_id*`)`
Gets the parent books of the given `id`.

Parameters **`book_id`** (`osid.id.Id`) – the `Id` of the `Book` to query

Returns the parent books of the `id`

Return type `osid.commenting.BookList`

Raise `NotFound` – a `Book` identified by `Id is` not found

Raise `NullArgument` – `book_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`is_ancestor_of_book`**`(`*id_*, *book_id*`)`
Tests if an `Id` is an ancestor of a book.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`book_id`** (`osid.id.Id`) – the `Id` of a book

Returns `tru` e if this `id` is an ancestor of `book_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `book_id` is not found

Raise `NullArgument` – `id` or `book_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` not found return `false`.

`CommentingManager.`**`has_child_books`**(*book_id*)

Tests if a book has any children.

Parameters **`book_id`** (`osid.id.Id`) – a book `Id`

Returns `true` if the `book_id` has children, `false` otherwise

Return type `boolean`

Raise `NotFound` – `book_id` is not found

Raise `NullArgument` – `book_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`is_child_of_book`**(*id_*, *book_id*)

Tests if a book is a direct child of another.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`book_id`** (`osid.id.Id`) – the `Id` of a book

Returns `true` if the `id` is a child of `book_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `book_id` is not found

Raise `NullArgument` – `id` or `book_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` not found return `false`.

`CommentingManager.`**`get_child_book_ids`**(*book_id*)

Gets the child `Ids` of the given book.

Parameters **`book_id`** (`osid.id.Id`) – the `Id` to query

Returns the children of the book

Return type `osid.id.IdList`

Raise `NotFound` – `book_id` is not found

Raise `NullArgument` – `book_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`get_child_books`**(*book_id*)

Gets the child books of the given `id`.

Parameters **`book_id`** (`osid.id.Id`) – the `Id` of the `Book` to query

Returns the child books of the `id`

Return type `osid.commenting.BookList`

Raise `NotFound` – a `Book` identified by `Id is` not found

Raise `NullArgument` – `book_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`is_descendant_of_book`**(*id_*, *book_id*)

Tests if an `Id` is a descendant of a book.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`book_id`** (`osid.id.Id`) – the `Id` of a book

Returns `true` if the `id` is a descendant of the `book_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `book_id` is not found

Raise `NullArgument` – `id` or `book_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` is not found return `false`.

`CommentingManager.`**`get_book_node_ids`**(*book_id*, *ancestor_levels*, *descendant_levels*, *include_siblings*)

Gets a portion of the hierarchy for the given book.

Parameters

- **`book_id`** (`osid.id.Id`) – the `Id` to query
- **`ancestor_levels`** (`cardinal`) – the maximum number of ancestor levels to include. A value of 0 returns no parents in the node.
- **`descendant_levels`** (`cardinal`) – the maximum number of descendant levels to include. A value of 0 returns no children in the node.
- **`include_siblings`** (`boolean`) – `true` to include the siblings of the given node, `false` to omit the siblings

Returns a book node

Return type `osid.hierarchy.Node`

Raise `NotFound` – `book_id` is not found

Raise `NullArgument` – `book_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

CommentingManager.**get_book_nodes**(*book_id*, *ancestor_levels*, *descendant_levels*, *include_siblings*)

Gets a portion of the hierarchy for the given book.

Parameters

- **book_id** (osid.id.Id) – the Id to query
- **ancestor_levels** (cardinal) – the maximum number of ancestor levels to include. A value of 0 returns no parents in the node.
- **descendant_levels** (cardinal) – the maximum number of descendant levels to include. A value of 0 returns no children in the node.
- **include_siblings** (boolean) – true to include the siblings of the given node, false to omit the siblings

Returns a book node

Return type osid.commenting.BookNode

Raise NotFound – book_id is not found

Raise NullArgument – book_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Book Hierarchy Design Methods

CommentingManager.**book_hierarchy_id**

Gets the hierarchy Id associated with this session.

Returns the hierarchy Id associated with this session

Return type osid.id.Id

compliance: mandatory – This method must be implemented.

CommentingManager.**book_hierarchy**

Gets the hierarchy associated with this session.

Returns the hierarchy associated with this session

Return type osid.hierarchy.Hierarchy

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

CommentingManager.**can_modify_book_hierarchy**()

Tests if this user can change the hierarchy.

A return of true does not guarantee successful authorization. A return of false indicates that it is known performing any update will result in a PermissionDenied. This is intended as a hint to an application that may opt not to offer these operations to an unauthorized user.

Returns false if changing this hierarchy is not authorized, true otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`add_root_book`**(*book_id*)

Adds a root book.

Parameters **`book_id`** (`osid.id.Id`) – the `Id` of a book

Raise `AlreadyExists` – `book_id` is already in hierarchy

Raise `NotFound` – `book_id` is not found

Raise `NullArgument` – `book_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`remove_root_book`**(*book_id*)

Removes a root book.

Parameters **`book_id`** (`osid.id.Id`) – the `Id` of a book

Raise `NotFound` – `book_id` is not a root

Raise `NullArgument` – `book_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`add_child_book`**(*book_id*, *child_id*)

Adds a child to a book.

Parameters

- **`book_id`** (`osid.id.Id`) – the `Id` of a book
- **`child_id`** (`osid.id.Id`) – the `Id` of the new child

Raise `AlreadyExists` – `book_id` is already a parent of `child_id`

Raise `NotFound` – `book_id` or `child_id` not found

Raise `NullArgument` – `book_id` or `child_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`remove_child_book`**(*book_id*, *child_id*)

Removes a child from a book.

Parameters

- **`book_id`** (`osid.id.Id`) – the `Id` of a book
- **`child_id`** (`osid.id.Id`) – the `Id` of the new child

Raise `NotFound` – `book_id` not a parent of `child_id`

Raise `NullArgument` – `book_id` or `child_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`CommentingManager.`**`remove_child_books`**(*book_id*)

Removes all children from a book.

Parameters **`book_id`** (`osid.id.Id`) – the `Id` of a book

Raise `NotFound` – `book_id` not found

Raise `NullArgument` – `book_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Book

Book

class `dlkit.services.commenting.`**`Book`**(*provider_manager*, *catalog*, *runtime*, *proxy*, ***kwargs*)

Bases: *`dlkit.osid.objects.OsidCatalog`*, `dlkit.osid.sessions.OsidSession`

A `Book` represents a collection of comments.

Like all OSID objects, a `Book` is identified by its `Id` and any persisted references should use the `Id`.

`get_book_record`(*book_record_type*)

Gets the book record corresponding to the given `Book` record `Type`.

This method is used to retrieve an object implementing the requested record. The `book_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(book_record_type)` is `true` .

Parameters **`book_record_type`** (`osid.type.Type`) – the type of book record to retrieve

Returns the book record

Return type `osid.commenting.records.BookRecord`

Raise `NullArgument` – `book_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(book_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Comment Lookup Methods

`Book.`**`book_id`**

Gets the `Book Id` associated with this session.

Returns the `Book Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Book.`**`book`**

Gets the `Book` associated with this session.

Returns the book

Return type `osid.commenting.Book`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Book.`**`can_lookup_comments`**`()`

Tests if this user can examine this book.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer these operations.

Returns `false` if book reading methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Book.`**`use_comparative_comment_view`**`()`

The returns from the lookup methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`Book.`**`use_plenary_comment_view`**`()`

A complete view of the `Comment` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`Book.`**`use_federated_book_view`**`()`

Federates the view for methods in this session.

A federated view will include comments in books which are children of this book in the book hierarchy.

compliance: mandatory – This method is must be implemented.

`Book.`**`use_isolated_book_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts searches to this book only.

compliance: mandatory – This method is must be implemented.

`Book.`**`use_effective_comment_view`**`()`

Only comments whose effective dates are current are returned by methods in this session.

compliance: mandatory – This method is must be implemented.

`Book.`**`use_any_effective_comment_view`**`()`

All comments of any effective dates are returned by all methods in this session.

compliance: mandatory – This method is must be implemented.

`Book.`**`get_comment`**`(`*comment_id*`)`

Gets the `Comment` specified by its `Id`.

Parameters **comment_id** (`osid.id.Id`) – the `Id` of the `Comment` to retrieve

Returns the returned `Comment`

Return type `osid.commenting.Comment`

Raise `NotFound` – no `Comment` found with the given `Id`

Raise `NullArgument` – `comment_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Book.`**get_comments_by_ids**(*comment_ids*)

Gets a `CommentList` corresponding to the given `IdList`.

Parameters **comment_ids** (`osid.id.IdList`) – the list of `Ids` to retrieve

Returns the returned `Comment list`

Return type `osid.commenting.CommentList`

Raise `NotFound` – an `Id was` not found

Raise `NullArgument` – `comment_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Book.`**get_comments_by_genus_type**(*comment_genus_type*)

Gets a `CommentList` corresponding to the given comment genus `Type` which does not include comments of genus types derived from the specified `Type`.

Parameters **comment_genus_type** (`osid.type.Type`) – a comment genus type

Returns the returned `Comment` list

Return type `osid.commenting.CommentList`

Raise `NullArgument` – `comment_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Book.`**get_comments_by_parent_genus_type**(*comment_genus_type*)

Gets a `CommentList` corresponding to the given comment genus `Type` and include any additional comments with genus types derived from the specified `Type`.

Parameters **comment_genus_type** (`osid.type.Type`) – a comment genus type

Returns the returned `Comment` list

Return type `osid.commenting.CommentList`

Raise `NullArgument` – `comment_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Book.**get_comments_by_record_type**(*comment_record_type*)
Gets a CommentList containing the given comment record Type.

Parameters **comment_record_type** (osid.type.Type) – a comment record type

Returns the returned Comment list

Return type osid.commenting.CommentList

Raise NullArgument – comment_record_type is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Book.**get_comments_on_date**(*from_*, *to*)
Gets a CommentList effective during the entire given date range inclusive but not confined to the date range.

Parameters

- **from** (osid.calendaring.DateTime) – starting date
- **to** (osid.calendaring.DateTime) – ending date

Returns the returned Comment list

Return type osid.commenting.CommentList

Raise InvalidArgument – from is greater than to

Raise NullArgument – from or to is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Book.**get_comments_by_genus_type_on_date**(*comment_genus_type*, *from_*, *to*)
Gets a CommentList of a given genus type and effective during the entire given date range inclusive but not confined to the date range.

Parameters

- **comment_genus_type** (osid.type.Type) – a comment genus type
- **from** (osid.calendaring.DateTime) – starting date
- **to** (osid.calendaring.DateTime) – ending date

Returns the returned Comment list

Return type osid.commenting.CommentList

Raise InvalidArgument – from is greater than to

Raise NullArgument – comment_genus_type, from, or to is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Book.**get_comments_for_commentor**(*resource_id*)
Gets a list of comments corresponding to a resource Id.

Parameters **`resource_id`** (`osid.id.Id`) – the `Id` of the resource

Returns the returned `CommentList`

Return type `osid.commenting.CommentList`

Raise `NullArgument` – `resource_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Book.`**`get_comments_for_commentor_on_date`**(*resource_id*, *from_*, *to*)

Gets a list of all comments corresponding to a resource `Id` and effective during the entire given date range inclusive but not confined to the date range.

Parameters

- **`resource_id`** (`osid.id.Id`) – the `Id` of the resource
- **`from`** (`osid.calendaring.DateTime`) – from date
- **`to`** (`osid.calendaring.DateTime`) – to date

Returns the returned `CommentList`

Return type `osid.commenting.CommentList`

Raise `InvalidArgument` – `to` is less than `from`

Raise `NullArgument` – `resource_id, from,` or `to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Book.`**`get_comments_by_genus_type_for_commentor`**(*resource_id*, *comment_genus_type*)

Gets a list of comments of the given genus type corresponding to a resource `Id`.

Parameters

- **`resource_id`** (`osid.id.Id`) – the `Id` of the resource
- **`comment_genus_type`** (`osid.type.Type`) – the comment genus type

Returns the returned `CommentList`

Return type `osid.commenting.CommentList`

Raise `NullArgument` – `resource_id` or `comment_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Book.`**`get_comments_by_genus_type_for_commentor_on_date`**(*resource_id*, *comment_genus_type*, *from_*, *to*)

Gets a list of all comments of the given genus type corresponding to a resource `Id` and effective during the entire given date range inclusive but not confined to the date range.

Parameters

- **resource_id** (`osid.id.Id`) – the `Id` of the resource
- **comment_genus_type** (`osid.type.Type`) – the comment genus type
- **from** (`osid.calendaring.DateTime`) – from date
- **to** (`osid.calendaring.DateTime`) – to date

Returns the returned `CommentList`

Return type `osid.commenting.CommentList`

Raise `InvalidArgument` – `to` is less than `from`

Raise `NullArgument` – `resource_id, comment_genus_type, from,` or `to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Book.`**`get_comments_for_reference`**(*reference_id*)

Gets a list of comments corresponding to a reference `Id`.

Parameters **reference_id** (`osid.id.Id`) – the `Id` of the reference

Returns the returned `CommentList`

Return type `osid.commenting.CommentList`

Raise `NullArgument` – `reference_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Book.`**`get_comments_for_reference_on_date`**(*reference_id*, *from_*, *to*)

Gets a list of all comments corresponding to a reference `Id` and effective during the entire given date range inclusive but not confined to the date range.

Parameters

- **reference_id** (`osid.id.Id`) – a reference `Id`
- **from** (`osid.calendaring.DateTime`) – from date
- **to** (`osid.calendaring.DateTime`) – to date

Returns the returned `CommentList`

Return type `osid.commenting.CommentList`

Raise `InvalidArgument` – `to` is less than `from`

Raise `NullArgument` – `reference_id, from,` or `to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Book.`**`get_comments_by_genus_type_for_reference`**(*reference_id*, *comment_genus_type*)

Gets a list of comments of the given genus type corresponding to a reference `Id`.

Parameters

- **reference_id** (`osid.id.Id`) – the `Id` of the reference
- **comment_genus_type** (`osid.type.Type`) – the comment genus type

Returns the returned `CommentList`

Return type `osid.commenting.CommentList`

Raise `NullArgument` – `reference_id` or `comment_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Book.`**`get_comments_by_genus_type_for_reference_on_date`**(*reference_id*, *comment_genus_type*, *from_*, *to*)

Gets a list of all comments of the given genus type corresponding to a reference `Id` and effective during the entire given date range inclusive but not confined to the date range.

Parameters

- **from** (`osid.calendaring.DateTime`) – from date
- **reference_id** (`osid.id.Id`) – a reference `Id`
- **comment_genus_type** (`osid.type.Type`) – the comment genus type
- **to** (`osid.calendaring.DateTime`) – to date

Returns the returned `CommentList`

Return type `osid.commenting.CommentList`

Raise `InvalidArgument` – `to` is less than `from`

Raise `NullArgument` – `reference_id, comment_genus_type, from,` or `to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Book.`**`get_comments_for_commentor_and_reference`**(*resource_id*, *reference_id*)

Gets a list of comments corresponding to a resource and reference `Id`.

Parameters

- **resource_id** (`osid.id.Id`) – the `Id` of the resource
- **reference_id** (`osid.id.Id`) – the `Id` of the reference

Returns the returned `CommentList`

Return type `osid.commenting.CommentList`

Raise `NullArgument` – `resource_id` or `reference_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Book.**get_comments_for_commentor_and_reference_on_date**(*resource_id*, *reference_id*, *from_*, *to*)

Gets a list of all comments corresponding to a resource and reference `Id` and effective during the entire given date range inclusive but not confined to the date range.

Parameters

- **resource_id** (`osid.id.Id`) – the `Id` of the resource
- **reference_id** (`osid.id.Id`) – a reference `Id`
- **from** (`osid.calendaring.DateTime`) – from date
- **to** (`osid.calendaring.DateTime`) – to date

Returns the returned `CommentList`

Return type `osid.commenting.CommentList`

Raise `InvalidArgument` – `to` is less than `from`

Raise `NullArgument` – `resource_id, reference_id, from,` or `to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Book.**get_comments_by_genus_type_for_commentor_and_reference**(*resource_id*, *reference_id*, *comment_genus_type*)

Gets a list of comments of the given genus type corresponding to a resource and reference `Id`.

Parameters

- **resource_id** (`osid.id.Id`) – the `Id` of the resource
- **reference_id** (`osid.id.Id`) – the `Id` of the reference
- **comment_genus_type** (`osid.type.Type`) – the comment genus type

Returns the returned `CommentList`

Return type `osid.commenting.CommentList`

Raise `NullArgument` – `resource_id, reference_id` or `comment_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Book.**get_comments_by_genus_type_for_commentor_and_reference_on_date**(*resource_id*, *reference_id*, *comment_genus_type*, *from_*, *to*)

Gets a list of all comments corresponding to a resource and reference `Id` and effective during the entire given date range inclusive but not confined to the date range.

Parameters

- **`resource_id`** (`osid.id.Id`) – the `Id` of the resource
- **`reference_id`** (`osid.id.Id`) – a reference `Id`
- **`comment_genus_type`** (`osid.type.Type`) – the comment genus type
- **`from`** (`osid.calendaring.DateTime`) – from date
- **`to`** (`osid.calendaring.DateTime`) – to date

Returns the returned `CommentList`

Return type `osid.commenting.CommentList`

Raise `InvalidArgument` – `to` is less than `from`

Raise `NullArgument` – `resource_id, reference_id, comment_genus_type, from,` or `to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Book.`**`comments`**

Gets all comments.

Returns a list of comments

Return type `osid.commenting.CommentList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Comment Query Methods

`Book.`**`book_id`**

Gets the `Book Id` associated with this session.

Returns the `Book Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Book.`**`book`**

Gets the `Book` associated with this session.

Returns the book

Return type `osid.commenting.Book`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Book.`**`can_search_comments`**`()`

Tests if this user can perform comment searches.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer search operations to unauthorized users.

Returns `false` if search methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Book.`**`use_federated_book_view`**`()`

Federates the view for methods in this session.

A federated view will include comments in books which are children of this book in the book hierarchy.

compliance: mandatory – This method is must be implemented.

`Book.`**`use_isolated_book_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts searches to this book only.

compliance: mandatory – This method is must be implemented.

`Book.`**`comment_query`**

Gets a comment query.

Returns the comment query

Return type `osid.commenting.CommentQuery`

compliance: mandatory – This method must be implemented.

`Book.`**`get_comments_by_query`**`(`*comment_query*`)`

Gets a list of comments matching the given search.

Parameters **`comment_query`** (`osid.commenting.CommentQuery`) – the search query array

Returns the returned `CommentList`

Return type `osid.commenting.CommentList`

Raise `NullArgument` – `comment_query` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `comment_query` is not of this service

compliance: mandatory – This method must be implemented.

Comment Admin Methods

`Book.`**`book_id`**

Gets the `Book Id` associated with this session.

Returns the `Book Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Book.`**`book`**

Gets the `Book` associated with this session.

Returns the book

Return type `osid.commenting.Book`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Book.`**`can_create_comments`**`()`

Tests if this user can create comments.

A return of true does not guarantee successful authorization. A return of false indicates that it is known creating a `Comment` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer create operations to unauthorized users.

Returns `false` if `Comment` creation is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Book.`**`can_create_comment_with_record_types`**`(`*comment_record_types*`)`

Tests if this user can create a single `Comment` using the desired record types.

While `CommentingManager.getCommentRecordTypes()` can be used to examine which records are supported, this method tests which record(s) are required for creating a specific `Comment`. Providing an empty array tests if a `Comment` can be created with no records.

Parameters **`comment_record_types`** (`osid.type.Type[]`) – array of comment record types

Returns `true` if `Comment` creation using the specified record `Types` is supported, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `comment_record_types` is `null`

compliance: mandatory – This method must be implemented.

`Book.`**`get_comment_form_for_create`**`(`*reference_id*, *comment_record_types*`)`

Gets the comment form for creating new comments.

A new form should be requested for each create transaction.

Parameters

- **`reference_id`** (`osid.id.Id`) – the `Id` for the reference object
- **`comment_record_types`** (`osid.type.Type[]`) – array of comment record types

Returns the comment form

Return type `osid.commenting.CommentForm`

Raise `NullArgument` – `reference_id or comment_record_types` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – unable to get form for requested record types

compliance: mandatory – This method must be implemented.

`Book.`**`create_comment`**`(`*comment_form*`)`

Creates a new `Comment`.

Parameters **`comment_form`** (`osid.commenting.CommentForm`) – the form for this `Comment`

Returns the new `Comment`

Return type `osid.commenting.Comment`

Raise `IllegalState` – `comment_form` already used in a create transaction

Raise `InvalidArgument` – one or more of the form elements is invalid

Raise `NullArgument` – `comment_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `comment_form` did not originate from `get_comment_form_for_create()`

compliance: mandatory – This method must be implemented.

`Book.`**`can_update_comments`**`()`

Tests if this user can update comments.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating a `Comment` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer update operations to unauthorized users.

Returns `false` if `Comment` modification is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Book.`**`get_comment_form_for_update`**`(`*comment_id*`)`

Gets the comment form for updating an existing comment.

A new comment form should be requested for each update transaction.

Parameters **`comment_id`** (`osid.id.Id`) – the `Id` of the `Comment`

Returns the comment form

Return type `osid.commenting.CommentForm`

Raise `NotFound` – `comment_id` is not found

Raise `NullArgument` – `comment_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Book.`**`update_comment`**`(`*comment_form*`)`

Updates an existing comment.

Parameters **`comment_form`** (`osid.commenting.CommentForm`) – the form containing the elements to be updated

Raise `IllegalState` – `comment_form` already used in an update transaction

Raise `InvalidArgument` – the form contains an invalid value

Raise `NullArgument` – `comment_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `comment_form` did not originate from `get_comment_form_for_update()`

compliance: mandatory – This method must be implemented.

`Book.`**`can_delete_comments`**`()`

Tests if this user can delete comments.

A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting an `Comment` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer delete operations to unauthorized users.

Returns `false` if `Comment` deletion is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Book.`**`delete_comment`**`(`*comment_id*`)`

Deletes a `Comment`.

Parameters **`comment_id`** (`osid.id.Id`) – the `Id` of the `Comment` to remove

Raise `NotFound` – `comment_id` not found

Raise `NullArgument` – `comment_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Book.`**`can_manage_comment_aliases`**`()`

Tests if this user can manage `Id` aliases for `Comnents`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known changing an alias will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer alias operations to an unauthorized user.

Returns `false` if `Comment` aliasing is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Book.`**`alias_comment`**`(`*comment_id*, *alias_id*`)`

Adds an `Id` to a `Comment` for the purpose of creating compatibility.

The primary `Id` of the `Comment` is determined by the provider. The new `Id` performs as an alias to the primary `Id`. If the alias is a pointer to another comment, it is reassigned to the given comment `Id`.

Parameters

- **`comment_id`** (`osid.id.Id`) – the `Id` of a `Comment`
- **`alias_id`** (`osid.id.Id`) – the alias `Id`

Raise `AlreadyExists` – `alias_id` is already assigned

Raise `NotFound` – `comment_id` not found

Raise `NullArgument` – `comment_id` or `alias_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Objects

Comment

class `dlkit.commenting.objects.`**`Comment`**

Bases: *`dlkit.osid.objects.OsidRelationship`*

A `Comment` represents a comment and/or rating related to a reference object in a book.

`reference_id`

Gets the `Id` of the referenced object to which this comment pertains.

Returns the reference `Id`

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`commentor_id`

Gets the `Id` of the resource who created this comment.

Returns the `Resource Id`

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`commentor`

Gets the resource who created this comment.

Returns the `Resource`

Return type `osid.resource.Resource`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`commenting_agent_id`

Gets the `Id` of the agent who created this comment.

Returns the `Agent Id`

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`commenting_agent`

Gets the agent who created this comment.

Returns the `Agent`

Return type `osid.authentication.Agent`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`text`

Gets the comment text.

Returns the comment text

Return type `osid.locale.DisplayText`

compliance: mandatory – This method must be implemented.

`has_rating``()`

Tests if this comment includes a rating.

Returns `true` if this comment includes a rating, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`rating_id`

Gets the `Id` of the `Grade`.

Returns the `Agent Id`

Return type `osid.id.Id`

Raise `IllegalState` – `has_rating()` is `false`

compliance: mandatory – This method must be implemented.

`rating`

Gets the `Grade`.

Returns the `Grade`

Return type `osid.grading.Grade`

Raise `IllegalState` – `has_rating()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_comment_record``(`*comment_record_type*`)`

Gets the comment record corresponding to the given `Comment` record `Type`.

This method is used to retrieve an object implementing the requested record. The `comment_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(comment_record_type)` is `true` .

Parameters **`comment_record_type`** (`osid.type.Type`) – the type of comment record to retrieve

Returns the comment record

Return type `osid.commenting.records.CommentRecord`

Raise `NullArgument` – `comment_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(comment_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Comment Form

class `dlkit.commenting.objects.`**`CommentForm`**

Bases: *`dlkit.osid.objects.OsidRelationshipForm`*

This is the form for creating and updating `Comment` objects.

Like all `OsidForm` objects, various data elements may be set here for use in the create and update methods in the `CommentAdminSession`. For each data element that may be set, metadata may be examined to provide display hints or data constraints.

`text_metadata`

Gets the metadata for the text.

Returns metadata for the text

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`text`

`rating_metadata`

Gets the metadata for a rating.

Returns metadata for the rating

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`rating`

`get_comment_form_record`(*comment_record_type*)

Gets the `CommentFormRecord` corresponding to the given comment record `Type`.

Parameters **`comment_record_type`** (`osid.type.Type`) – the comment record type

Returns the comment form record

Return type `osid.commenting.records.CommentFormRecord`

Raise `NullArgument` – `comment_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(comment_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Comment List

class `dlkit.commenting.objects.`**`CommentList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists, CommentList` provides a means for accessing `Comment` elements sequentially either one at a time or many at a time.

Examples: while (cl.hasNext()) { Comment comment = cl.getNextComment(); }

or

while (cl.hasNext()) { Comment[] comments = cl.getNextComments(cl.available());

}

next_comment

Gets the next Comment in this list.

Returns the next Comment in this list. The has_next() method should be used to test that a next Comment is available before calling this method.

Return type osid.commenting.Comment

Raise IllegalState – no more elements available in this list

Raise OperationFailed – unable to complete request

compliance: mandatory – This method must be implemented.

get_next_comments(*n*)

Gets the next set of Comment elements in this list.

The specified amount must be less than or equal to the return from available().

Parameters **n** (cardinal) – the number of Comment elements requested which must be less than or equal to available()

Returns an array of Comment elements.The length of the array is less than or equal to the number specified.

Return type osid.commenting.Comment

Raise IllegalState – no more elements available in this list

Raise OperationFailed – unable to complete request

compliance: mandatory – This method must be implemented.

Book

class dlkit.commenting.objects.**Book**(*abc_commenting_objects.Book*, *osid_objects.OsidCatalog*)
:noindex:

get_book_record(*book_record_type*)

Gets the book record corresponding to the given Book record Type.

This method is used to retrieve an object implementing the requested record. The book_record_type may be the Type returned in get_record_types() or any of its parents in a Type hierarchy where has_record_type(book_record_type) is true .

Parameters **book_record_type** (osid.type.Type) – the type of book record to retrieve

Returns the book record

Return type osid.commenting.records.BookRecord

Raise NullArgument – book_record_type is null

Raise OperationFailed – unable to complete request

Raise Unsupported – has_record_type(book_record_type) is false

compliance: mandatory – This method must be implemented.

Book Form

class `dlkit.commenting.objects.`**`BookForm`**

Bases: *`dlkit.osid.objects.OsidCatalogForm`*

This is the form for creating and updating `Books`.

Like all `OsidForm` objects, various data elements may be set here for use in the create and update methods in the `BookAdminSession`. For each data element that may be set, metadata may be examined to provide display hints or data constraints.

`get_book_form_record`(*book_record_type*)

Gets the `BookFormRecord` corresponding to the given book record `Type`.

Parameters **`book_record_type`** (`osid.type.Type`) – the book record type

Returns the book form record

Return type `osid.commenting.records.BookFormRecord`

Raise `NullArgument` – `book_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(book_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Book List

class `dlkit.commenting.objects.`**`BookList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists, BookList` provides a means for accessing `Book` elements sequentially either one at a time or many at a time.

Examples: while (bl.hasNext()) { Book book = bl.getNextBook(); }

or

while (bl.hasNext()) { Book[] books = bl.getNextBooks(bl.available());

}

`next_book`

Gets the next `Book` in this list.

Returns the next `Book` in this list. The `has_next()` method should be used to test that a next `Book` is available before calling this method.

Return type `osid.commenting.Book`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_books`(*n*)

Gets the next set of `Book` elements in this list.

The specified amount must be less than or equal to the return from `available()`.

Parameters **`n`** (`cardinal`) – the number of `Book` elements requested which must be less than or equal to `available()`

Returns an array of `Book` elements.The length of the array is less than or equal to the number specified.

Return type `osid.commenting.Book`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Book Node

class `dlkit.commenting.objects.`**`BookNode`**

Bases: *`dlkit.osid.objects.OsidNode`*

This interface is a container for a partial hierarchy retrieval.

The number of hierarchy levels traversable through this interface depend on the number of levels requested in the `BookHierarchySession`.

`book`

Gets the `Book` at this node.

Returns the book represented by this node

Return type `osid.commenting.Book`

compliance: mandatory – This method must be implemented.

`parent_book_nodes`

Gets the parents of this book.

Returns the parents of this book

Return type `osid.commenting.BookNodeList`

compliance: mandatory – This method must be implemented.

`child_book_nodes`

Gets the children of this book.

Returns the children of this book

Return type `osid.commenting.BookNodeList`

compliance: mandatory – This method must be implemented.

Book Node List

class `dlkit.commenting.objects.`**`BookNodeList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists, BookNodeList` provides a means for accessing `BookNode` elements sequentially either one at a time or many at a time.

Examples: while (bnl.hasNext()) { BookNode node = bnl.getNextBookNode(); }

or

while (bnl.hasNext()) { BookNode[] nodes = bnl.getNextBookNodes(bnl.available());

}

next_book_node

Gets the next `BookNode` in this list.

Returns the next `BookNode` in this list. The `has_next()` method should be used to test that a next `BookNode` is available before calling this method.

Return type `osid.commenting.BookNode`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

get_next_book_nodes(*n*)

Gets the next set of `BookNode` elements in this list.

The specified amount must be less than or equal to the return from `available()`.

Parameters n (`cardinal`) – the number of `BookNode` elements requested which must be less than or equal to `available()`

Returns an array of `BookNode` elements.The length of the array is less than or equal to the number specified.

Return type `osid.commenting.BookNode`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Queries

Comment Query

class `dlkit.commenting.queries.`**CommentQuery**

Bases: *`dlkit.osid.queries.OsidRelationshipQuery`*

This is the query for searching comments.

Each method specifies an `AND` term while multiple invocations of the same method produce a nested `OR`.

match_reference_id(*source_id*, *match*)

Sets reference `Id`.

Parameters

- **source_id** (`osid.id.Id`) – a source `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `source_id` is `null`

compliance: mandatory – This method must be implemented.

reference_id_terms

match_commentor_id(*resource_id*, *match*)

Sets a resource `Id` to match a commentor.

Parameters

- **resource_id** (`osid.id.Id`) – a resource `Id`

- **match** (boolean) – true for a positive match, false for a negative match

Raise NullArgument – resource_id is null

compliance: mandatory – This method must be implemented.

commentor_id_terms

supports_commentor_query()
Tests if a ResourceQuery is available.

Returns true if a resource query is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

commentor_query
Gets the query for a resource query.

Multiple retrievals produce a nested OR term.

Returns the resource query

Return type osid.resource.ResourceQuery

Raise Unimplemented – supports_commentor_query() is false

compliance: optional – This method must be implemented if ``supports_commentor_query()`` is ``true``.

commentor_terms

match_commenting_agent_id(*agent_id*, *match*)
Sets an agent Id.

Parameters

- **agent_id** (osid.id.Id) – an agent Id
- **match** (boolean) – true for a positive match, false for a negative match

Raise NullArgument – agent_id is null

compliance: mandatory – This method must be implemented.

commenting_agent_id_terms

supports_commenting_agent_query()
Tests if an AgentQuery is available.

Returns true if an agent query is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

commenting_agent_query
Gets the query for an agent query.

Multiple retrievals produce a nested OR term.

Returns the agent query

Return type osid.authentication.AgentQuery

Raise Unimplemented – supports_commenting_agent_query() is false

compliance: optional – This method must be implemented if ``supports_commenting_agent_query()`` is ``true``.

commenting_agent_terms

match_text(*text*, *string_match_type*, *match*)
Matches text.

Parameters

- **text** (`string`) – the text
- **string_match_type** (`osid.type.Type`) – a string match type
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `text` is not of `string_match_type`

Raise `NullArgument` – `text` is `null`

Raise `Unsupported` – `supports_string_match_type(string_match_type)` is `false`

compliance: mandatory – This method must be implemented.

match_any_text(*match*)
Matches a comment that has any text assigned.

Parameters **match** (`boolean`) – `true` to match comments with any text, `false` to match comments with no text

compliance: mandatory – This method must be implemented.

text_terms

match_rating_id(*grade_id*, *match*)
Sets a grade `Id`.

Parameters

- **grade_id** (`osid.id.Id`) – a grade `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `grade_id` is `null`

compliance: mandatory – This method must be implemented.

rating_id_terms

supports_rating_query()
Tests if a `GradeQuery` is available.

Returns `true` if a rating query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

rating_query
Gets the query for a rating query.

Multiple retrievals produce a nested `OR` term.

Returns the rating query

Return type `osid.grading.GradeQuery`

Raise `Unimplemented` – `supports_rating_query()` is `false`

compliance: optional – This method must be implemented if ``supports_rating_query()`` is ``true``.

match_any_rating(*match*)
Matches books with any rating.

Parameters **match** (`boolean`) – `true` to match comments with any rating, `false` to match comments with no ratings

compliance: mandatory – This method must be implemented.

rating_terms

match_book_id(*book_id*, *match*)
Sets the book `Id` for this query to match comments assigned to books.

Parameters

- **book_id** (`osid.id.Id`) – a book `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `book_id` is `null`

compliance: mandatory – This method must be implemented.

book_id_terms

supports_book_query()
Tests if a `BookQuery` is available.

Returns `true` if a book query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

book_query
Gets the query for a book query.

Multiple retrievals produce a nested `OR` term.

Returns the book query

Return type `osid.commenting.BookQuery`

Raise `Unimplemented` – `supports_book_query()` is `false`

compliance: optional – This method must be implemented if ``supports_book_query()`` is ``true``.

book_terms

get_comment_query_record(*comment_record_type*)
Gets the comment query record corresponding to the given `Comment` record `Type`.

Multiple record retrievals produce a nested `OR` term.

Parameters **comment_record_type** (`osid.type.Type`) – a comment record type

Returns the comment query record

Return type `osid.commenting.records.CommentQueryRecord`

Raise `NullArgument` – `comment_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(comment_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Book Query

class `dlkit.commenting.queries.`**`BookQuery`**

Bases: *`dlkit.osid.queries.OsidCatalogQuery`*

This is the query for searching books.

Each method specifies an `AND` term while multiple invocations of the same method produce a nested `OR`.

`match_comment_id`(*comment_id*, *match*)

Sets the comment `Id` for this query to match comments assigned to books.

Parameters

- **`comment_id`** (`osid.id.Id`) – a comment `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `comment_id` is `null`

compliance: mandatory – This method must be implemented.

`comment_id_terms`

`supports_comment_query`()

Tests if a comment query is available.

Returns `true` if a comment query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`comment_query`

Gets the query for a comment.

Returns the comment query

Return type `osid.commenting.CommentQuery`

Raise `Unimplemented` – `supports_comment_query()` is `false`

compliance: optional – This method must be implemented if ``supports_comment_query()`` is ``true``.

`match_any_comment`(*match*)

Matches books with any comment.

Parameters **`match`** (`boolean`) – `true` to match books with any comment, `false` to match books with no comments

compliance: mandatory – This method must be implemented.

`comment_terms`

`match_ancestor_book_id`(*book_id*, *match*)

Sets the book `Id` for this query to match books that have the specified book as an ancestor.

Parameters

- **`book_id`** (`osid.id.Id`) – a book `Id`
- **`match`** (`boolean`) – `true` for a positive match, a `false` for a negative match

Raise `NullArgument` – `book_id` is `null`

compliance: mandatory – This method must be implemented.

`ancestor_book_id_terms`

supports_ancestor_book_query()
Tests if a BookQuery is available.

Returns true if a book query is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

ancestor_book_query
Gets the query for a book.

Multiple retrievals produce a nested OR term.

Returns the book query

Return type osid.commenting.BookQuery

Raise Unimplemented – supports_ancestor_book_query() is false

compliance: optional – This method must be implemented if ‘‘supports_ancestor_book_query()‘‘ is ‘‘true‘‘.

match_any_ancestor_book(*match*)
Matches books with any ancestor.

Parameters **match** (boolean) – true to match books with any ancestor, false to match root books

compliance: mandatory – This method must be implemented.

ancestor_book_terms

match_descendant_book_id(*book_id*, *match*)
Sets the book Id for this query to match books that have the specified book as a descendant.

Parameters

- **book_id** (osid.id.Id) – a book Id
- **match** (boolean) – true for a positive match, false for a negative match

Raise NullArgument – book_id is null

compliance: mandatory – This method must be implemented.

descendant_book_id_terms

supports_descendant_book_query()
Tests if a BookQuery is available.

Returns true if a book query is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

descendant_book_query
Gets the query for a book.

Multiple retrievals produce a nested OR term.

Returns the book query

Return type osid.commenting.BookQuery

Raise Unimplemented – supports_descendant_book_query() is false *compliance: optional – This method must be implemented if ‘‘supports_descendant_book_query()‘‘ is ‘‘true‘‘.*

match_any_descendant_book(*match*)
Matches books with any descendant.

Parameters **match** (boolean) – true to match books with any descendant, false to match leaf books

compliance: mandatory – This method must be implemented.

descendant_book_terms

get_book_query_record(*book_record_type*)
Gets the book query record corresponding to the given Book record Type.

Multiple record retrievals produce a nested boolean OR term.

Parameters **book_record_type** (osid.type.Type) – a book record type

Returns the book query record

Return type osid.commenting.records.BookQueryRecord

Raise NullArgument – book_record_type is null

Raise OperationFailed – unable to complete request

Raise Unsupported – has_record_type(book_record_type) is false

compliance: mandatory – This method must be implemented.

Records

Comment Record

class dlkit.commenting.records.**CommentRecord**
Bases: *dlkit.osid.records.OsidRecord*

A record for a Comment.

The methods specified by the record type are available through the underlying object.

Comment Query Record

class dlkit.commenting.records.**CommentQueryRecord**
Bases: *dlkit.osid.records.OsidRecord*

A record for a CommentQuery.

The methods specified by the record type are available through the underlying object.

Comment Form Record

class dlkit.commenting.records.**CommentFormRecord**
Bases: *dlkit.osid.records.OsidRecord*

A record for a CommentForm.

The methods specified by the record type are available through the underlying object.

Comment Search Record

class `dlkit.commenting.records.`**`CommentSearchRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `CommentSearch`.

The methods specified by the record type are available through the underlying object.

Book Record

class `dlkit.commenting.records.`**`BookRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `Book`.

The methods specified by the record type are available through the underlying object.

Book Query Record

class `dlkit.commenting.records.`**`BookQueryRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `BookQuery`.

The methods specified by the record type are available through the underlying object.

Book Form Record

class `dlkit.commenting.records.`**`BookFormRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `BookForm`.

The methods specified by the record type are available through the underlying object.

Book Search Record

class `dlkit.commenting.records.`**`BookSearchRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `BookSearch`.

The methods specified by the record type are available through the underlying object.

Grading

Summary

Grading Open Service Interface Definitions grading version 3.0.0

The Grading OSID defines a service to apply grades or ratings.

Grade Systems

The grade system sessions provide the means to retrievs and manage `GradeSystem` definitions. A `GradeSystem` is a fixed set of `Grades` . `GradeSystems` may also take the form of a numerical score as well as a rating based on some system. `GradeEntries` belong to a single `GradebookColumn`.

Gradebook Columns

A `Gradebook` is represented by a series of `GradebookColumns`. A `GradeBookColumn` represents a something to be graded and is joined to a single `GradeSystem`. A `GradebookColumn` may be constrained to a single grader.

Grade Entries

A `GradebookColumn` is comprised of a series of `GradeEntry` elements. A `GradebookColumn` may represent "Assignment 3" while a `GradeEntry` may represent the assignment turned in by a particular student.

A `Grade` can be applied to a `GradeEntry` that relates the entry to a grader and a key `Resource`. In the case of a class gradebook, the key resource represents the student. If there are multiple graders for the same key resource, each grader gets their own `GradebookColumn`.

Gradebooks may also be used to capture ratings about other objects. In the case where people vote for their favorite assets, the key resource represents the `Asset` .

`GradebookColumns` may have a `GradebookColumnSummary` entry for summary results and statistics across all `GradeEntries` in the column.

Gradebook Cataloging

`GradebookColumns` are organized into `Gradebooks`. `Gradebooks` also provide for a federated hierarchy of `GradebookColumns`. Simple reordering of `GradebookColumns` can be performed by moving the `GradebookColumn` relative to another. The relative positioning may reference two `GradebookColumns` through the federation.

Sub Packages

The Grading OSID includes several subpackages. The Grading Transform OSID provides a means of translating one `GradeSystem` to another. The Grading Calculation OSID defines derived `GradebookColumns`. The Grading Batch OSID manages `GradeSystems, GradeEntries, Gradebooks,` and `GradebookColumns` in bulk.
Grading Open Service Interface Definitions grading version 3.0.0

The Grading OSID defines a service to apply grades or ratings.

Grade Systems

The grade system sessions provide the means to retrievs and manage `GradeSystem` definitions. A `GradeSystem` is a fixed set of `Grades` . `GradeSystems` may also take the form of a numerical score as well as a rating based on some system. `GradeEntries` belong to a single `GradebookColumn`.

Gradebook Columns

A `Gradebook` is represented by a series of `GradebookColumns`. A `GradeBookColumn` represents a something to be graded and is joined to a single `GradeSystem`. A `GradebookColumn` may be constrained to a single grader.

Grade Entries

A `GradebookColumn` is comprised of a series of `GradeEntry` elements. A `GradebookColumn` may represent "Assignment 3" while a `GradeEntry` may represent the assignment turned in by a particular student.

A `Grade` can be applied to a `GradeEntry` that relates the entry to a grader and a key `Resource`. In the case of a class gradebook, the key resource represents the student. If there are multiple graders for the same key resource, each grader gets their own `GradebookColumn`.

Gradebooks may also be used to capture ratings about other objects. In the case where people vote for their favorite assets, the key resource represents the `Asset` .

`GradebookColumns` may have a `GradebookColumnSummary` entry for summary results and statistics across all `GradeEntries` in the column.

Gradebook Cataloging

`GradebookColumns` are organized into `Gradebooks`. `Gradebooks` also provide for a federated hierarchy of `GradebookColumns`. Simple reordering of `GradebookColumns` can be performed by moving the `GradebookColumn` relative to another. The relative positioning may reference two `GradebookColumns` through the federation.

Sub Packages

The Grading OSID includes several subpackages. The Grading Transform OSID provides a means of translating one `GradeSystem` to another. The Grading Calculation OSID defines derived `GradebookColumns`. The Grading Batch OSID manages `GradeSystems, GradeEntries, Gradebooks,` and `GradebookColumns` in bulk.

Service Managers

Grading Profile

class `dlkit.services.grading.`**`GradingProfile`**

Bases: `dlkit.osid.managers.OsidProfile`

The `GradingProfile` describes the interoperability among grading services.

`supports_grade_system_lookup``()`

Tests if a grade system lookup service is supported.

Returns true if grade system lookup is supported, false otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_grade_system_query``()`

Tests if a grade system query service is supported.

Returns `true` if grade system query is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_grade_system_admin``()`

Tests if a grade system administrative service is supported.

Returns `true` if grade system admin is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_grade_entry_lookup``()`

Tests if a grade entry lookup service is supported.

Returns true if grade entry lookup is supported, false otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_grade_entry_query``()`

Tests if a grade entry query service is supported.

Returns true if grade entry query is supported, false otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_grade_entry_admin()

Tests if a grade entry administrative service is supported.

Returns `true` if grade entry admin is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_gradebook_column_lookup()

Tests if a gradebook column lookup service is supported.

Returns true if gradebook column lookup is supported, false otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_gradebook_column_query()

Tests if a gradebook column query service is supported.

Returns `true` if grade system query is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_gradebook_column_admin()

Tests if a gradebook column administrative service is supported.

Returns `true` if gradebook column admin is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_gradebook_lookup()

Tests if a gradebook lookup service is supported.

Returns `true` if gradebook lookup is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_gradebook_admin()

Tests if a gradebook administrative service is supported.

Returns `true` if gradebook admin is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

grade_record_types

Gets the supported `Grade` record types.

Returns a list containing the supported `Grade` record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

grade_system_record_types

Gets the supported `GradeSystem` record types.

Returns a list containing the supported `GradeSystem` record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

grade_system_search_record_types

Gets the supported `GradeSystem` search record types.

Returns a list containing the supported `GradeSystem` search record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

grade_entry_record_types

Gets the supported `GradeEntry` record types.

Returns a list containing the supported `GradeEntry` record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

grade_entry_search_record_types

Gets the supported `GradeEntry` search record types.

Returns a list containing the supported `GradeEntry` search record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

gradebook_column_record_types

Gets the supported `GradebookColumn` record types.

Returns a list containing the supported `GradebookColumn` record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

gradebook_column_search_record_types

Gets the supported gradebook column search record types.

Returns a list containing the supported `GradebookColumn` search record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

gradebook_column_summary_record_types

Gets the supported `GradebookColumnSummary` record types.

Returns a list containing the supported `GradebookColumnSummary` record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

gradebook_record_types

Gets the supported `Gradebook` record types.

Returns a list containing the supported `Gradebook` record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

gradebook_search_record_types

Gets the supported gradebook search record types.

Returns a list containing the supported `Gradebook` search record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

Grading Manager

class `dlkit.services.grading.`**`GradingManager`**(*proxy=None*)

Bases: `dlkit.osid.managers.OsidManager`, `dlkit.osid.sessions.OsidSession`, *`dlkit.services.grading.GradingProfile`*

The grading manager provides access to grading sessions and provides interoperability tests for various aspects of this service.

The sessions included in this manager are:

- `GradeSystemLookupSession:` a session to look up grades and grade systems
- `GradeSystemQuerySession:` a session to query grade systems `None`
- `GradeSystemSearchSession:` a session to search grade systems
- `GradeSystemAdminSession:` a session to manage grade systems
- `GradeSystemNotificationSession` a session for subscribing to new or deleted grades or grade systems
- `GradeSystemGradebookSession:` a session for retrieving grade system to gradebook mappings
- `GradeSystemGradebookAssignmentSession:` a session for managing grade system to gradebook mappings
- `GradeSystemSmartGradebookSession:` a session for managing smart gradebooks of grade systems
- `GradeEntryLookupSession:` a session to look up grade entries
- `GradeEntryQuerySession:` a session to query grade entries `None`
- `GradeEntrySearchSession:` a session to search grade entries
- `GradeEntryAdminSession:` a session to create, modify and delete grade entries `None`
- `GradeEntryNotificationSession: a` session to receive messages pertaining to grade entry ```` changes
- `GradebookColumnLookupSession:` a session to look up gradebook columns
- `GradebookColumnQuerySession:` a session to query gradebook columns `None`
- `GradebookColumnSearchSession:` a session to search gradebook columns
- `GradebookColumnAdminSession:` a session to manage gradebook columns
- `GradebookColumnNotificationSession` a session for subscribing to new or deleted gradebook columns
- `GradebookColumnGradebookSession:` a session for retrieving gradebook column to gradebook mappings

- `GradebookColumnGradebookAssignmentSession`: a session for managing gradebook column to gradebook mappings
- `GradebookColumnSmartGradebookSession`: a session for managing smart gradebooks of gradebook columns
- `GradebookLookupSession`: a session to lookup gradebooks
- `GradebookQuerySession`: a session to query gradebooks
- `GradebookSearchSession` : a session to search gradebooks
- `GradebookAdminSession` : a session to create, modify and delete gradebooks
- `GradebookNotificationSession` : a session to receive messages pertaining to gradebook changes
- `GradebookHierarchySession`: a session to traverse the gradebook hierarchy
- `GradebookHierarchyDesignSession`: a session to manage the gradebook hierarchy

grading_batch_manager

Gets the `GradingBatchManager`.

Returns a `GradingBatchManager`

Return type `osid.grading.batch.GradingBatchManager`

Raise `OperationFailed` – unable to complete request

Raise `Unimplemented` – `supports_grading_batch() is false`

compliance: optional – This method must be implemented if ‘‘supports_grading_batch()‘‘ is true.

grading_calculation_manager

Gets the `GradingCalculationManager`.

Returns a `GradingCalculationManager`

Return type `osid.grading.calculation.GradingCalculationManager`

Raise `OperationFailed` – unable to complete request

Raise `Unimplemented` – `supports_grading_calculation() is false`

compliance: optional – This method must be implemented if ‘‘supports_grading_calculation()‘‘ is true.

grading_transform_manager

Gets the `GradingTransformManager`.

Returns a `GradingTransformManager`

Return type `osid.grading.transform.GradingTransformManager`

Raise `OperationFailed` – unable to complete request

Raise `Unimplemented` – `supports_grading_transform() is false`

compliance: optional – This method must be implemented if ‘‘supports_grading_transform()‘‘ is true.

Gradebook Column Lookup Methods

`GradingManager.`**gradebook_id**

Gets the `Gradebook Id` associated with this session.

Returns the `Gradebook Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`GradingManager.`**`gradebook`**

Gets the `Gradebook` associated with this session.

Returns the `Gradebook` associated with this session

Return type `osid.grading.Gradebook`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`GradingManager.`**`can_lookup_gradebook_columns`**`()`

Tests if this user can perform `GradebookColumn` lookups.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if lookup methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`GradingManager.`**`use_comparative_gradebook_column_view`**`()`

The returns from the lookup methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`GradingManager.`**`use_plenary_gradebook_column_view`**`()`

A complete view of the `GradebookColumn` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`GradingManager.`**`use_federated_gradebook_view`**`()`

Federates the view for methods in this session.

A federated view will include gradebook columns in gradebooks which are children of this gradebook in the gradebook hierarchy.

compliance: mandatory – This method is must be implemented.

`GradingManager.`**`use_isolated_gradebook_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts searches to this gradebook only.

compliance: mandatory – This method is must be implemented.

`GradingManager.`**`get_gradebook_column`**`(`*gradebook_column_id*`)`

Gets the `GradebookColumn` specified by its `Id`.

In plenary mode, the exact `Id` is found or a `NotFound` results. Otherwise, the returned `GradebookColumn` may have a different `Id` than requested, such as the case where a duplicate `Id` was assigned to a `GradebookColumn` and retained for compatibility.

Parameters **`gradebook_column_id`** (`osid.id.Id`) – `Id` of the `GradebookColumn`

Returns the gradebook column

Return type `osid.grading.GradebookColumn`

Raise `NotFound` – `gradebook_column_id` not found

Raise `NullArgument` – `gradebook_column_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method is must be implemented.

`GradingManager.`**`get_gradebook_columns_by_ids`**(*gradebook_column_ids*)

Gets a `GradebookColumnList` corresponding to the given `IdList`.

In plenary mode, the returned list contains all of the gradebook columns specified in the `Id` list, in the order of the list, including duplicates, or an error results if a `Id` in the supplied list is not found or inaccessible. Otherwise, inaccessible gradeboook columns may be omitted from the list.

Parameters **`gradebook_column_ids`** (`osid.id.IdList`) – the list of `Ids` to retrieve

Returns the returned `GradebookColumn` list

Return type `osid.grading.GradebookColumnList`

Raise `NotFound` – an `Id was` not found

Raise `NullArgument` – `grade_book_column_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`GradingManager.`**`get_gradebook_columns_by_genus_type`**(*gradebook_column_genus_type*)

Gets a `GradebookColumnList` corresponding to the given gradebook column genus `Type` which does not include gradebook columns of genus types derived from the specified `Type`.

In plenary mode, the returned list contains all known gradebook columns or an error results. Otherwise, the returned list may contain only those gradebook columns that are accessible through this session.

Parameters **`gradebook_column_genus_type`** (`osid.type.Type`) – a gradebook column genus type

Returns the returned `GradebookColumn` list

Return type `osid.grading.GradebookColumnList`

Raise `NullArgument` – `gradebook_column_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`GradingManager.`**`get_gradebook_columns_by_parent_genus_type`**(*gradebook_column_genus_type*)

Gets a `GradebookColumnList` corresponding to the given gradebook column genus `Type` and include any additional columns with genus types derived from the specified `Type`.

In plenary mode, the returned list contains all known gradebook columns or an error results. Otherwise, the returned list may contain only those gradebook columns that are accessible through this session.

Parameters **`gradebook_column_genus_type`** (`osid.type.Type`) – a gradebook column genus type

Returns the returned `GradebookColumn` list

Return type `osid.grading.GradebookColumnList`

Raise `NullArgument` – `gradebook_column_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`GradingManager.`**`get_gradebook_columns_by_record_type`**(*gradebook_column_record_type*)

Gets a `GradebookColumnList` containing the given gradebook column record `Type`.

In plenary mode, the returned list contains all known gradebook columns or an error results. Otherwise, the returned list may contain only those gradebook columns that are accessible through this session.

Parameters **`gradebook_column_record_type`** (`osid.type.Type`) – a gradebook column record type

Returns the returned `GradebookColumn` list

Return type `osid.grading.GradebookColumnList`

Raise `NullArgument` – `gradebook_column_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`GradingManager.`**`gradebook_columns`**

Gets all gradebook columns.

In plenary mode, the returned list contains all known gradebook columns or an error results. Otherwise, the returned list may contain only those gradebook columns that are accessible through this session.

Returns a `GradebookColumn`

Return type `osid.grading.GradebookColumnList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`GradingManager.`**`supports_summary`**()

Tests if a summary entry is available.

Returns `true` if a summary entry is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

GradingManager.**get_gradebook_column_summary**(*gradebook_column_id*)

Gets the GradebookColumnSummary for summary results.

Parameters **gradebook_column_id** (osid.id.Id) – Id of the GradebookColumn

Returns the gradebook column summary

Return type osid.grading.GradebookColumnSummary

Raise NotFound – gradebook_column_id is not found

Raise NullArgument – gradebook_column_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

Raise Unimplemented – has_summary() is false

compliance: mandatory – This method is must be implemented.

Gradebook Column Query Methods

GradingManager.**gradebook_id**

Gets the Gradebook Id associated with this session.

Returns the Gradebook Id associated with this session

Return type osid.id.Id

compliance: mandatory – This method must be implemented.

GradingManager.**gradebook**

Gets the Gradebook associated with this session.

Returns the Gradebook associated with this session

Return type osid.grading.Gradebook

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

GradingManager.**can_search_gradebook_columns**()

Tests if this user can perform GradebookColumn searches.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a PermissionDenied. This is intended as a hint to an application that may opt not to offer search operations to unauthorized users.

Returns false if search methods are not authorized, true otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

GradingManager.**use_federated_gradebook_view**()

Federates the view for methods in this session.

A federated view will include gradebook columns in gradebooks which are children of this gradebook in the gradebook hierarchy.

compliance: mandatory – This method is must be implemented.

`GradingManager.`**`use_isolated_gradebook_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts searches to this gradebook only.

compliance: mandatory – This method is must be implemented.

`GradingManager.`**`gradebook_column_query`**

Gets a gradebook column query.

Returns the gradebook column

Return type `osid.grading.GradebookColumnQuery`

compliance: mandatory – This method must be implemented.

`GradingManager.`**`get_gradebook_columns_by_query`**`(`*gradebook_column_query*`)`

Gets a list of gradebook columns matching the given query.

Parameters **`gradebook_column_query`** (`osid.grading.GradebookColumnQuery`) – the gradebook column query

Returns the returned `GradebookColumnList`

Return type `osid.grading.GradebookColumnList`

Raise `NullArgument` – `gradebook_column_query` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `gradebook_column_query` is not of this service

compliance: mandatory – This method must be implemented.

Gradebook Column Admin Methods

`GradingManager.`**`gradebook_id`**

Gets the `Gradebook Id` associated with this session.

Returns the `Gradebook Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`GradingManager.`**`gradebook`**

Gets the `Gradebook` associated with this session.

Returns the `Gradebook` associated with this session

Return type `osid.grading.Gradebook`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`GradingManager.`**`can_create_gradebook_columns`**`()`

Tests if this user can create gradebook columns.

A return of true does not guarantee successful authorization. A return of false indicates that it is known creating a gradebook column will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer create operations to an unauthorized user.

Returns `false` if `GradebookColumn` creation is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`GradingManager.`**`can_create_gradebook_column_with_record_types`**(*gradebook_column_record_types*)

Tests if this user can create a single `GradebookColumn` using the desired record types.

While `GradingManager.getGradebookColumnRecordTypes()` can be used to examine which records are supported, this method tests which record(s) are required for creating a specific `GradebookColumn`. Providing an empty array tests if a `GradebookColumn` can be created with no records.

Parameters **`gradebook_column_record_types`** (`osid.type.Type[]`) – array of gradebook column record types

Returns `true` if `GradebookColumn` creation using the specified record `Types` is supported, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `gradebook_column_record_types` is `null`

compliance: mandatory – This method must be implemented.

`GradingManager.`**`get_gradebook_column_form_for_create`**(*gradebook_column_record_types*)

Gets the gradebook column form for creating new gradebook columns.

A new form should be requested for each create transaction.

Parameters **`gradebook_column_record_types`** (`osid.type.Type[]`) – array of gradebook column record types

Returns the gradebook column form

Return type `osid.grading.GradebookColumnForm`

Raise `NullArgument` – `gradebook_column_record_types` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – unable to get form for requested record types

compliance: mandatory – This method must be implemented.

`GradingManager.`**`create_gradebook_column`**(*gradebook_column_form*)

Creates a new `GradebookColumn`.

Parameters **`gradebook_column_form`** (`osid.grading.GradebookColumnForm`) – the form for this `GradebookColumn`

Returns the new `GradebookColumn`

Return type `osid.grading.GradebookColumn`

Raise `IllegalState` – `gradebook_column_form` already used in a create transaction

Raise `InvalidArgument` – one or more of the form elements is invalid

Raise `NullArgument` – `gradebook_column_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `gradebook_column_form` did not originate from `get_gradebook_column_form_for_create()`

compliance: mandatory – This method must be implemented.

`GradingManager.`**`can_update_gradebook_columns`**`()`

Tests if this user can update gradebook columns.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating a `GradebookColumn` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer update operations to an unauthorized user.

Returns `false` if gradebook column modification is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`GradingManager.`**`get_gradebook_column_form_for_update`**`(`*gradebook_column_id*`)`

Gets the gradebook column form for updating an existing gradebook column.

A new gradebook column form should be requested for each update transaction.

Parameters **`gradebook_column_id`** (`osid.id.Id`) – the `Id` of the `GradebookColumn`

Returns the gradebook column form

Return type `osid.grading.GradebookColumnForm`

Raise `NotFound` – `gradebook_column_id` is not found

Raise `NullArgument` – `gradebook_column_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`GradingManager.`**`update_gradebook_column`**`(`*gradebook_column_form*`)`

Updates an existing gradebook column.

Parameters **`gradebook_column_form`** (`osid.grading.GradebookColumnForm`) – the form containing the elements to be updated

Raise `IllegalState` – `gradebook_column_form` already used in an update transaction

Raise `InvalidArgument` – the form contains an invalid value

Raise `NullArgument` – `gradebook_column_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `gradebook_column_form` did not originate from `get_gradebook_column_form_for_update()`

compliance: mandatory – This method must be implemented.

`GradingManager.`**`sequence_gradebook_columns`**`(`*gradebook_column_ids*`)`

Resequences the gradebook columns.

Parameters **`gradebook_column_ids`** (`osid.id.IdList`) – the `Ids` of the `GradebookColumns`

Raise `NullArgument` – `gradebook_column_id_list` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`GradingManager.`**`move_gradebook_column`**(*front_gradebook_column_id*, *back_gradebook_column_id*)

Moves a gradebook column in front of another.

Parameters

- **`front_gradebook_column_id`** (`osid.id.Id`) – the `Id` of a `GradebookColumn`
- **`back_gradebook_column_id`** (`osid.id.Id`) – the `Id` of a `GradebookColumn`

Raise `NotFound` – `front_gradebook_column_id or back_gradebook_column_id` is not found

Raise `NullArgument` – `front_gradebook_column_id or back_gradebook_column_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`GradingManager.`**`copy_gradebook_column_entries`**(*source_gradebook_column_id*, *target_gradebook_column_id*)

Copies gradebook column entries from one column to another.

If the target grade column grade system differs from the source, the grades in the entries are transformed to the new grade system.

Parameters

- **`source_gradebook_column_id`** (`osid.id.Id`) – the `Id` of a `GradebookColumn`
- **`target_gradebook_column_id`** (`osid.id.Id`) – the `Id` of a `GradebookColumn`

Raise `NotFound` – `source_gradebook_column_id ortarget_gradebook_column_id` is not found

Raise `NullArgument` – `source_gradebook_column_id target_gradebook_column_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`GradingManager.`**`can_delete_gradebook_columns`**()

Tests if this user can delete gradebook columns.

A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting a `GradebookColumn` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer delete operations to an unauthorized user.

Returns `false` if `GradebookColumn` deletion is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`GradingManager.`**`delete_gradebook_column`**`(`*gradebook_column_id*`)`

Deletes the `GradebookColumn` identified by the given `Id`.

Parameters **`gradebook_column_id`** (`osid.id.Id`) – the `Id` of the `GradebookColumn` to delete

Raise `NotFound` – a `GradebookColumn` was not found identified by the given `Id`

Raise `NullArgument` – `gradebook_column_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`GradingManager.`**`can_manage_gradebook_column_aliases`**`()`

Tests if this user can manage `Id` aliases for `GradebookColumns`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known changing an alias will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer alias operations to an unauthorized user.

Returns `false` if `GradebookColumn` aliasing is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`GradingManager.`**`alias_gradebook_column`**`(`*gradebook_column_id, alias_id*`)`

Adds an `Id` to a `GradebookColumn` for the purpose of creating compatibility.

The primary `Id` of the `GradebookColumn` is determined by the provider. The new `Id` performs as an alias to the primary `Id`. If the alias is a pointer to another gradebook column, it is reassigned to the given gradebook column `Id`.

Parameters

- **`gradebook_column_id`** (`osid.id.Id`) – the `Id` of a `GradebookColumn`
- **`alias_id`** (`osid.id.Id`) – the alias `Id`

Raise `AlreadyExists` – `alias_id` is already assigned

Raise `NotFound` – `gradebook_column_id` not found

Raise `NullArgument` – `gradebook_column_id` or `alias_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Gradebook Lookup Methods

`GradingManager.`**`can_lookup_gradebooks`**`()`

Tests if this user can perform `Gradebook` lookups.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if lookup methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`GradingManager.`**`use_comparative_gradebook_view`**`()`

The returns from the lookup methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`GradingManager.`**`use_plenary_gradebook_view`**`()`

A complete view of the `Gradebook` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`GradingManager.`**`get_gradebook`**`(`*gradebook_id*`)`

Gets the `Gradebook` specified by its `Id`.

In plenary mode, the exact `Id` is found or a `NotFound` results. Otherwise, the returned `Gradebook` may have a different `Id` than requested, such as the case where a duplicate `Id` was assigned to a `Gradebook` and retained for compatility.

Parameters **`gradebook_id`** (`osid.id.Id`) – `Id` of the `Gradebook`

Returns the gradebook

Return type `osid.grading.Gradebook`

Raise `NotFound` – `gradebook_id` not found

Raise `NullArgument` – `gradebook_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method is must be implemented.

`GradingManager.`**`get_gradebooks_by_ids`**`(`*gradebook_ids*`)`

Gets a `GradebookList` corresponding to the given `IdList`.

In plenary mode, the returned list contains all of the gradebooks specified in the `Id` list, in the order of the list, including duplicates, or an error results if an `Id` in the supplied list is not found or inaccessible. Otherwise, inaccessible `Gradebook` objects may be omitted from the list and may present the elements in any order including returning a unique set.

Parameters **`gradebook_ids`** (`osid.id.IdList`) – the list of `Ids` to retrieve

Returns the returned `Gradebook` list

Return type `osid.grading.GradebookList`

Raise `NotFound` – an `Id was` not found

Raise `NullArgument` – `gradebook_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

GradingManager.**get_gradebooks_by_genus_type**(*gradebook_genus_type*)

Gets a GradebookList corresponding to the given gradebook genus Type which does not include gradebooks of types derived from the specified Type.

In plenary mode, the returned list contains all known gradebooks or an error results. Otherwise, the returned list may contain only those gradebooks that are accessible through this session.

Parameters **gradebook_genus_type** (osid.type.Type) – a gradebook genus type

Returns the returned Gradebook list

Return type osid.grading.GradebookList

Raise NullArgument – gradebook_genus_type is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

GradingManager.**get_gradebooks_by_parent_genus_type**(*gradebook_genus_type*)

Gets a GradebookList corresponding to the given gradebook genus Type and include any additional gradebooks with genus types derived from the specified Type.

In plenary mode, the returned list contains all known gradebooks or an error results. Otherwise, the returned list may contain only those gradebooks that are accessible through this session.

Parameters **gradebook_genus_type** (osid.type.Type) – a gradebook genus type

Returns the returned Gradebook list

Return type osid.grading.GradebookList

Raise NullArgument – gradebook_genus_type is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

GradingManager.**get_gradebooks_by_record_type**(*gradebook_record_type*)

Gets a GradebookList containing the given gradebook record Type.

In plenary mode, the returned list contains all known gradebooks or an error results. Otherwise, the returned list may contain only those gradebooks that are accessible through this session.

Parameters **gradebook_record_type** (osid.type.Type) – a gradebook record type

Returns the returned Gradebook list

Return type osid.grading.GradebookList

Raise NullArgument – gradebook_record_type is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

GradingManager.**get_gradebooks_by_provider**(*resource_id*)

Gets a GradebookList for the given provider ““.

In plenary mode, the returned list contains all known gradebooks or an error results. Otherwise, the returned list may contain only those gradebooks that are accessible through this session.

Parameters **resource_id** (osid.id.Id) – a resource Id

Returns the returned Gradebook list

Return type osid.grading.GradebookList

Raise NullArgument – resource_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

GradingManager.**gradebooks**

Gets all Gradebooks.

In plenary mode, the returned list contains all known gradebooks or an error results. Otherwise, the returned list may contain only those gradebooks that are accessible through this session.

Returns a GradebookList

Return type osid.grading.GradebookList

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Gradebook Admin Methods

GradingManager.**can_create_gradebooks**()

Tests if this user can create Gradebooks.

A return of true does not guarantee successful authorization. A return of false indicates that it is known creating a Gradebook will result in a PermissionDenied. This is intended as a hint to an application that may not wish to offer create operations to unauthorized users.

Returns false if Gradebook creation is not authorized, true otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

GradingManager.**can_create_gradebook_with_record_types**(*gradebook_record_types*)

Tests if this user can create a single Gradebook using the desired record types.

While GradingManager.getGradebookRecordTypes() can be used to examine which records are supported, this method tests which record(s) are required for creating a specific Gradebook. Providing an empty array tests if a Gradebook can be created with no records.

Parameters **gradebook_record_types** (osid.type.Type[]) – array of gradebook record types

Returns true if Gradebook creation using the specified Types is supported, false otherwise

Return type boolean

Raise `NullArgument` – `gradebook_record_types` is `null`

compliance: mandatory – This method must be implemented.

`GradingManager.`**`get_gradebook_form_for_create`**`(`*gradebook_record_types*`)`

Gets the gradebook form for creating new gradebooks.

A new form should be requested for each create transaction.

Parameters **`gradebook_record_types`** (`osid.type.Type[]`) – array of gradebook record types

Returns the gradebook form

Return type `osid.grading.GradebookForm`

Raise `NullArgument` – `gradebook_record_types` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – unable to get form for requested record types

compliance: mandatory – This method must be implemented.

`GradingManager.`**`create_gradebook`**`(`*gradebook_form*`)`

Creates a new `Gradebook`.

Parameters **`gradebook_form`** (`osid.grading.GradebookForm`) – the form for this `Gradebook`

Returns the new `Gradebook`

Return type `osid.grading.Gradebook`

Raise `IllegalState` – `gradebook_form` already used in a create transaction

Raise `InvalidArgument` – one or more of the form elements is invalid

Raise `NullArgument` – `gradebook_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `gradebook_form` did not originate from `get_gradebook_form_for_create()`

compliance: mandatory – This method must be implemented.

`GradingManager.`**`can_update_gradebooks`**`()`

Tests if this user can update `Gradebooks`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating a `Gradebook` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer update operations to unauthorized users.

Returns `false` if `Gradebook` modification is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`GradingManager.`**`get_gradebook_form_for_update`**`(`*gradebook_id*`)`

Gets the gradebook form for updating an existing gradebook.

A new gradebook form should be requested for each update transaction.

Parameters **gradebook_id** (`osid.id.Id`) – the `Id` of the `Gradebook`

Returns the gradebook form

Return type `osid.grading.GradebookForm`

Raise `NotFound` – `gradebook_id` is not found

Raise `NullArgument` – `gradebook_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`GradingManager.`**`update_gradebook`**`(`*gradebook_form*`)`

Updates an existing gradebook.

Parameters **gradebook_form** (`osid.grading.GradebookForm`) – the form containing the elements to be updated

Raise `IllegalState` – `gradebook_form` already used in an update transaction

Raise `InvalidArgument` – the form contains an invalid value

Raise `NullArgument` – `gradebook_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `gradebook_form did not originate from get_gradebook_form_for_update()`

compliance: mandatory – This method must be implemented.

`GradingManager.`**`can_delete_gradebooks`**`()`

Tests if this user can delete gradebooks.

A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting a `Gradebook` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer delete operations to unauthorized users.

Returns `false` if `Gradebook` deletion is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`GradingManager.`**`delete_gradebook`**`(`*gradebook_id*`)`

Deletes a `Gradebook`.

Parameters **gradebook_id** (`osid.id.Id`) – the `Id` of the `Gradebook` to remove

Raise `NotFound` – `gradebook_id` not found

Raise `NullArgument` – `gradebook_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`GradingManager.`**`can_manage_gradebook_aliases`**`()`

Tests if this user can manage `Id` aliases for `Gradebooks`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known changing an alias will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer alias operations to an unauthorized user.

Returns `false` if `Gradebook` aliasing is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`GradingManager.`**`alias_gradebook`**`(`*gradebook_id*, *alias_id*`)`

Adds an `Id` to a `Gradebook` for the purpose of creating compatibility.

The primary `Id` of the `Gradebook` is determined by the provider. The new `Id` performs as an alias to the primary `Id` . If the alias is a pointer to another gradebook, it is reassigned to the given gradebook `Id`.

Parameters

- **`gradebook_id`** (`osid.id.Id`) – the `Id` of a `Gradebook`
- **`alias_id`** (`osid.id.Id`) – the alias `Id`

Raise `AlreadyExists` – `alias_id` is already assigned

Raise `NotFound` – `gradebook_id` not found

Raise `NullArgument` – `gradebook_id` or `alias_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Gradebook

Gradebook

class `dlkit.services.grading.`**`Gradebook`**`(`*provider_manager*, *catalog*, *runtime*, *proxy*, ***kwargs*`)`

Bases: *`dlkit.osid.objects.OsidCatalog`*, `dlkit.osid.sessions.OsidSession`

A gradebook defines a collection of grade entries.

`get_gradebook_record``(`*gradebook_record_type*`)`

Gets the gradebook record corresponding to the given `Gradebook` record `Type`.

This method is used to retrieve an object implementing the requested record. The `gradebook_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(gradebook_record_type)` is `true` .

Parameters **`gradebook_record_type`** (`osid.type.Type`) – a gradebook record type

Returns the gradebook record

Return type `osid.grading.records.GradebookRecord`

Raise `NullArgument` – `gradebook_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(gradebook_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Grade System Lookup Methods

`Gradebook.`**`gradebook_id`**

Gets the `Gradebook Id` associated with this session.

Returns the `Gradebook Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`gradebook`**

Gets the `Gradebook` associated with this session.

Returns the `Gradebook` associated with this session

Return type `osid.grading.Gradebook`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Gradebook.`**`can_lookup_grade_systems`**`()`

Tests if this user can perform `GradeSystem` lookups.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if lookup methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`use_comparative_grade_system_view`**`()`

The returns from the lookup methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`Gradebook.`**`use_plenary_grade_system_view`**`()`

A complete view of the `GradeSystem` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`Gradebook.`**`use_federated_gradebook_view`**`()`

Federates the view for methods in this session.

A federated view will include grade entries in gradebooks which are children of this gradebook in the gradebook hierarchy.

compliance: mandatory – This method is must be implemented.

`Gradebook.`**`use_isolated_gradebook_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts searches to this gradebook only.

compliance: mandatory – This method is must be implemented.

`Gradebook.`**`get_grade_system`**(*grade_system_id*)

Gets the `GradeSystem` specified by its `Id`.

In plenary mode, the exact `Id` is found or a `NotFound` results. Otherwise, the returned `GradeSystem` may have a different `Id` than requested, such as the case where a duplicate `Id` was assigned to a `GradeSystem` and retained for compatibility.

Parameters **`grade_system_id`** (`osid.id.Id`) – `Id` of the `GradeSystem`

Returns the grade system

Return type `osid.grading.GradeSystem`

Raise `NotFound` – `grade_system_id` not found

Raise `NullArgument` – `grade_system_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method is must be implemented.

`Gradebook.`**`get_grade_system_by_grade`**(*grade_id*)

Gets the `GradeSystem` by a `Grade Id`.

Parameters **`grade_id`** (`osid.id.Id`) – `Id` of a `Grade`

Returns the grade system

Return type `osid.grading.GradeSystem`

Raise `NotFound` – `grade_id` not found

Raise `NullArgument` – `grade_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method is must be implemented.

`Gradebook.`**`get_grade_systems_by_ids`**(*grade_system_ids*)

Gets a `GradeSystemList` corresponding to the given `IdList`.

In plenary mode, the returned list contains all of the systems specified in the `Id` list, in the order of the list, including duplicates, or an error results if an `Id` in the supplied list is not found or inaccessible. Otherwise, inaccessible `GradeSystems` may be omitted from the list and may present the elements in any order including returning a unique set.

Parameters **`grade_system_ids`** (`osid.id.IdList`) – the list of `Ids` to retrieve

Returns the returned `GradeSystem` list

Return type `osid.grading.GradeSystemList`

Raise `NotFound` – an `Id was` not found

Raise `NullArgument` – `grade_system_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Gradebook.`**`get_grade_systems_by_genus_type`**(*grade_system_genus_type*)

Gets a `GradeSystemList` corresponding to the given grade system genus `Type` which does not include systems of genus types derived from the specified `Type`.

In plenary mode, the returned list contains all known systems or an error results. Otherwise, the returned list may contain only those systems that are accessible through this session.

Parameters **`grade_system_genus_type`** (`osid.type.Type`) – a grade system genus type

Returns the returned `GradeSystem` list

Return type `osid.grading.GradeSystemList`

Raise `NullArgument` – `grade_system_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Gradebook.`**`get_grade_systems_by_parent_genus_type`**(*grade_system_genus_type*)

Gets a `GradeSystemList` corresponding to the given grade system genus `Type` and include any additional systems with genus types derived from the specified `Type`.

In plenary mode, the returned list contains all known systems or an error results. Otherwise, the returned list may contain only those systems that are accessible through this session.

Parameters **`grade_system_genus_type`** (`osid.type.Type`) – a grade system genus type

Returns the returned `GradeSystem` list

Return type `osid.grading.GradeSystemList`

Raise `NullArgument` – `grade_system_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Gradebook.`**`get_grade_systems_by_record_type`**(*grade_system_record_type*)

Gets a `GradeSystemList` containing the given grade record `Type`.

In plenary mode, the returned list contains all known systems or an error results. Otherwise, the returned list may contain only those systems that are accessible through this session.

Parameters **`grade_system_record_type`** (`osid.type.Type`) – a grade system record type

Returns the returned `GradeSystem` list

Return type `osid.grading.GradeSystemList`

Raise `NullArgument` – `grade_system_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Gradebook.`**`grade_systems`**

Gets all `GradeSystems`.

In plenary mode, the returned list contains all known grade systems or an error results. Otherwise, the returned list may contain only those grade systems that are accessible through this session.

Returns a `GradeSystemList`

Return type `osid.grading.GradeSystemList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Grade System Query Methods

`Gradebook.`**`gradebook_id`**

Gets the `Gradebook Id` associated with this session.

Returns the `Gradebook Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`gradebook`**

Gets the `Gradebook` associated with this session.

Returns the `Gradebook` associated with this session

Return type `osid.grading.Gradebook`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Gradebook.`**`can_search_grade_systems`**`()`

Tests if this user can perform `GradeSystem` searches.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer search operations to unauthorized users.

Returns `false` if search methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`use_federated_gradebook_view`**`()`

Federates the view for methods in this session.

A federated view will include grade entries in gradebooks which are children of this gradebook in the gradebook hierarchy.

compliance: mandatory – This method is must be implemented.

`Gradebook.`**`use_isolated_gradebook_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts searches to this gradebook only.

compliance: mandatory – This method is must be implemented.

`Gradebook.`**`grade_system_query`**

Gets a grade system query.

Returns a grade system query

Return type `osid.grading.GradeSystemQuery`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`get_grade_systems_by_query`**`(`*grade_system_query*`)`

Gets a list of `GradeSystem` objects matching the given grade system query.

Parameters **`grade_system_query`** (`osid.grading.GradeSystemQuery`) – the grade system query

Returns the returned `GradeSystemList`

Return type `osid.grading.GradeSystemList`

Raise `NullArgument` – `grade_system_query` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `grade_system_query` is not of this service

compliance: mandatory – This method must be implemented.

Grade System Admin Methods

`Gradebook.`**`gradebook_id`**

Gets the `Gradebook Id` associated with this session.

Returns the `Gradebook Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`gradebook`**

Gets the `Gradebook` associated with this session.

Returns the `Gradebook` associated with this session

Return type `osid.grading.Gradebook`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Gradebook.`**`can_create_grade_systems`**`()`

Tests if this user can create `GradeSystems`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known creating a `GradeSystem` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer create operations to unauthorized users.

Returns `false` if `GradeSystem` creation is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`can_create_grade_system_with_record_types`**(*grade_system_record_types*)

Tests if this user can create a single `GradeSystem` using the desired record types.

While `GradingManager.getGradeSystemRecordTypes()` can be used to examine which records are supported, this method tests which record(s) are required for creating a specific `GradeSystem`. Providing an empty array tests if a `GradeSystem` can be created with no records.

Parameters **`grade_system_record_types`** (`osid.type.Type[]`) – array of grade system types

Returns `true` if `GradeSystem` creation using the specified `Types` is supported, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `grade_system_record_types` is `null`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`get_grade_system_form_for_create`**(*grade_system_record_types*)

Gets the grade system form for creating new grade systems.

A new form should be requested for each create transaction.

Parameters **`grade_system_record_types`** (`osid.type.Type[]`) – array of grade system types

Returns the grade system form

Return type `osid.grading.GradeSystemForm`

Raise `NullArgument` – `grade_system_record_types` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – unable to get form for requested record types

compliance: mandatory – This method must be implemented.

`Gradebook.`**`create_grade_system`**(*grade_system_form*)

Creates a new `GradeSystem`.

Parameters **`grade_system_form`** (`osid.grading.GradeSystemForm`) – the form for this `GradeSystem`

Returns the new `GradeSystem`

Return type `osid.grading.GradeSystem`

Raise `IllegalState` – `grade_system_form` already used in a create transaction

Raise `InvalidArgument` – one or more of the form elements is invalid

Raise `NullArgument` – `grade_system_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `grade_system_form` did not originate from `get_grade_system_form_for_create()`

compliance: mandatory – This method must be implemented.

Gradebook.**can_update_grade_systems**()

Tests if this user can update GradeSystems.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating a GradeSystem will result in a PermissionDenied. This is intended as a hint to an application that may not wish to offer update operations to unauthorized users.

Returns false if GradeSystem modification is not authorized, true otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

Gradebook.**get_grade_system_form_for_update**(*grade_system_id*)

Gets the grade system form for updating an existing grade system.

A new grade system form should be requested for each update transaction.

Parameters **grade_system_id** (osid.id.Id) – the Id of the GradeSystem

Returns the grade system form

Return type osid.grading.GradeSystemForm

Raise NotFound – grade_system_id is not found

Raise NullArgument – grade_system_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Gradebook.**update_grade_system**(*grade_system_form*)

Updates an existing grade system.

Parameters **grade_system_form** (osid.grading.GradeSystemForm) – the form containing the elements to be updated

Raise IllegalState – grade_system_form already used in an update transaction

Raise InvalidArgument – the form contains an invalid value

Raise NullArgument – grade_system_form is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

Raise Unsupported – grade_system_form did not originate from get_grade_system_form_for_update()

compliance: mandatory – This method must be implemented.

Gradebook.**can_delete_grade_systems**()

Tests if this user can delete grade systems.

A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting a GradeSystem will result in a PermissionDenied. This is intended as a hint to an application that may not wish to offer delete operations to unauthorized users.

Returns false if GradeSystem deletion is not authorized, true otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

Gradebook.**delete_grade_system**(*grade_system_id*)

Deletes a GradeSystem.

Parameters **grade_system_id** (osid.id.Id) – the Id of the GradeSystem to remove

Raise NotFound – grade_system_id not found

Raise NullArgument – grade_system_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Gradebook.**can_manage_grade_system_aliases**()

Tests if this user can manage Id aliases for GradeSystems.

A return of true does not guarantee successful authorization. A return of false indicates that it is known changing an alias will result in a PermissionDenied. This is intended as a hint to an application that may opt not to offer alias operations to an unauthorized user.

Returns false if GradeSystem aliasing is not authorized, true otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

Gradebook.**alias_grade_system**(*grade_system_id*, *alias_id*)

Adds an Id to a GradeSystem for the purpose of creating compatibility.

The primary Id of the GradeSystem is determined by the provider. The new Id performs as an alias to the primary Id. If the alias is a pointer to another grade system, it is reassigned to the given grade system Id.

Parameters

- **grade_system_id** (osid.id.Id) – the Id of a GradeSystem
- **alias_id** (osid.id.Id) – the alias Id

Raise AlreadyExists – alias_id is already assigned

Raise NotFound – grade_system_id not found

Raise NullArgument – grade_system_id or alias_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Gradebook.**can_create_grades**(*grade_system_id*)

Tests if this user can create Grade s for a GradeSystem.

A return of true does not guarantee successful authorization. A return of false indicates that it is known creating a GradeSystem will result in a PermissionDenied. This is intended as a hint to an application that may not wish to offer create operations to unauthorized users.

Parameters **grade_system_id** (osid.id.Id) – the Id of a GradeSystem

Returns false if Grade creation is not authorized, true otherwise

Return type boolean

Raise NullArgument – grade_system_id is null

compliance: mandatory – This method must be implemented.

`Gradebook.`**`can_create_grade_with_record_types`**(*grade_system_id*, *grade_record_types*)

Tests if this user can create a single `Grade` using the desired record types.

While `GradingManager.getGradeRecordTypes()` can be used to examine which records are supported, this method tests which record(s) are required for creating a specific `Grade`. Providing an empty array tests if a `Grade` can be created with no records.

Parameters

- **`grade_system_id`** (`osid.id.Id`) – the `Id` of a `GradeSystem`
- **`grade_record_types`** (`osid.type.Type[]`) – array of grade recod types

Returns `true` if `Grade` creation using the specified `Types` is supported, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `grade_system_id` or `grade_record_types` is `null`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`get_grade_form_for_create`**(*grade_system_id*, *grade_record_types*)

Gets the grade form for creating new grades.

A new form should be requested for each create transaction.

Parameters

- **`grade_system_id`** (`osid.id.Id`) – the `Id` of a `GradeSystem`
- **`grade_record_types`** (`osid.type.Type[]`) – array of grade recod types

Returns the grade form

Return type `osid.grading.GradeForm`

Raise `NotFound` – `grade_system_id` is not found

Raise `NullArgument` – `grade_system_id` or `grade_record_types` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – unable to get form for requested record types

compliance: mandatory – This method must be implemented.

`Gradebook.`**`create_grade`**(*grade_form*)

Creates a new `Grade`.

Parameters **`grade_form`** (`osid.grading.GradeForm`) – the form for this `Grade`

Returns the new `Grade`

Return type `osid.grading.Grade`

Raise `IllegalState` – `grade_form` already used in a create transaction

Raise `InvalidArgument` – one or more of the form elements is invalid

Raise `NullArgument` – `grade_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `grade_form` did not originate from `get_grade_form_for_create()`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`can_update_grades`**`(`*grade_system_id*`)`

Tests if this user can update `Grades`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating a `Grade` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer update operations to unauthorized users.

Parameters **`grade_system_id`** (`osid.id.Id`) – the `Id` of a `GradeSystem`

Returns `false` if `Grade` modification is not authorized, `true` otherwise

Return type `boolean`

Raise `NullArgument` – `grade_system_id` is `null`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`get_grade_form_for_update`**`(`*grade_id*`)`

Gets the grade form for updating an existing grade.

A new grade form should be requested for each update transaction.

Parameters **`grade_id`** (`osid.id.Id`) – the `Id` of the `Grade`

Returns the grade form

Return type `osid.grading.GradeForm`

Raise `NotFound` – `grade_id` is not found

Raise `NullArgument` – `grade_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Gradebook.`**`update_grade`**`(`*grade_form*`)`

Updates an existing grade.

Parameters **`grade_form`** (`osid.grading.GradeForm`) – the form containing the elements to be updated

Raise `IllegalState` – `grade_form` already used in an update transaction

Raise `InvalidArgument` – the form contains an invalid value

Raise `NullArgument` – `grade_id` or `grade_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `grade_form` did not originate from `get_grade_form_for_update()`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`can_delete_grades`**`(`*grade_system_id*`)`

Tests if this user can delete grades.

A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting a `Grade` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer delete operations to unauthorized users.

Parameters **`grade_system_id`** (`osid.id.Id`) – the `Id` of a `GradeSystem`

Returns `false` if `Grade` deletion is not authorized, `true` otherwise

Return type `boolean`

Raise `NullArgument` – `grade_system_id` is `null`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`delete_grade`**`(`*grade_id*`)`

Deletes a `Grade`.

Parameters **`grade_id`** (`osid.id.Id`) – the `Id` of the `Grade` to remove

Raise `NotFound` – `grade_id` not found

Raise `NullArgument` – `grade_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Gradebook.`**`can_manage_grade_aliases`**`()`

Tests if this user can manage `Id` aliases for `Grades`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known changing an alias will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer alias operations to an unauthorized user.

Returns `false` if `Grade` aliasing is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`alias_grade`**`(`*grade_id*, *alias_id*`)`

Adds an `Id` to a `Grade` for the purpose of creating compatibility.

The primary `Id` of the `Grade` is determined by the provider. The new `Id` performs as an alias to the primary `Id`. If the alias is a pointer to another grade, it is reassigned to the given grade `Id`.

Parameters

- **`grade_id`** (`osid.id.Id`) – the `Id` of a `Grade`
- **`alias_id`** (`osid.id.Id`) – the alias `Id`

Raise `AlreadyExists` – `alias_id` is already assigned

Raise `NotFound` – `grade_id` not found

Raise `NullArgument` – `grade_id` or `alias_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Grade Entry Lookup Methods

`Gradebook.`**`gradebook_id`**

Gets the `Gradebook Id` associated with this session.

Returns the `Gradebook Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`gradebook`**

Gets the `Gradebook` associated with this session.

Returns the `Gradebook` associated with this session

Return type `osid.grading.Gradebook`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Gradebook.`**`can_lookup_grade_entries`**`()`

Tests if this user can perform `GradeEntry` lookups.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if lookup methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`use_comparative_grade_entry_view`**`()`

The returns from the lookup methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`Gradebook.`**`use_plenary_grade_entry_view`**`()`

A complete view of the `GradeEntry` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`Gradebook.`**`use_federated_gradebook_view`**`()`

Federates the view for methods in this session.

A federated view will include grade entries in gradebooks which are children of this gradebook in the gradebook hierarchy.

compliance: mandatory – This method is must be implemented.

`Gradebook.`**`use_isolated_gradebook_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts searches to this gradebook only.

compliance: mandatory – This method is must be implemented.

Gradebook.**use_effective_grade_entry_view**()

Only grade entries whose effective dates are current are returned by methods in this session.

compliance: mandatory – This method is must be implemented.

Gradebook.**use_any_effective_grade_entry_view**()

All grade entries of any effective dates are returned by methods in this session.

compliance: mandatory – This method is must be implemented.

Gradebook.**get_grade_entry**(*grade_entry_id*)

Gets the `GradeEntry` specified by its `Id`.

Parameters **grade_entry_id** (`osid.id.Id`) – `Id` of the `GradeEntry`

Returns the grade entry

Return type `osid.grading.GradeEntry`

Raise `NotFound` – `grade_entry_id` not found

Raise `NullArgument` – `grade_entry_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method is must be implemented.

Gradebook.**get_grade_entries_by_ids**(*grade_entry_ids*)

Gets a `GradeEntryList` corresponding to the given `IdList`.

Parameters **grade_entry_ids** (`osid.id.IdList`) – the list of `Ids` to retrieve

Returns the returned `GradeEntry` list

Return type `osid.grading.GradeEntryList`

Raise `NotFound` – an `Id was` not found

Raise `NullArgument` – `grade_entry_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Gradebook.**get_grade_entries_by_genus_type**(*grade_entry_genus_type*)

Gets a `GradeEntryList` corresponding to the given grade entry genus `Type` which does not include grade entries of genus types derived from the specified `Type`.

Parameters **grade_entry_genus_type** (`osid.type.Type`) – a grade entry genus type

Returns the returned `GradeEntry` list

Return type `osid.grading.GradeEntryList`

Raise `NullArgument` – `grade_entry_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Gradebook.**get_grade_entries_by_parent_genus_type**(*grade_entry_genus_type*)
Gets a GradeEntryList corresponding to the given grade entry genus Type and include any additional grade entry with genus types derived from the specified Type.

Parameters **grade_entry_genus_type** (osid.type.Type) – a grade entry genus type

Returns the returned GradeEntry list

Return type osid.grading.GradeEntryList

Raise NullArgument – grade_entry_genus_type is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Gradebook.**get_grade_entries_by_record_type**(*grade_entry_record_type*)
Gets a GradeEntryList containing the given grade entry record Type.

Parameters **grade_entry_record_type** (osid.type.Type) – a grade entry record type

Returns the returned GradeEntry list

Return type osid.grading.GradeEntryList

Raise NullArgument – grade_entry_record_type is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Gradebook.**get_grade_entries_on_date**(*from_*, *to*)
Gets a GradeEntryList effective during the entire given date range inclusive but not confined to the date range.

Parameters

- **from** (osid.calendaring.DateTime) – start of date range
- **to** (osid.calendaring.DateTime) – end of date range

Returns the returned GradeEntry list

Return type osid.grading.GradeEntryList

Raise InvalidArgument – from is greater than to

Raise NullArgument – from or to is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Gradebook.**get_grade_entries_for_gradebook_column**(*gradebook_column_id*)
Gets a GradeEntryList for the gradebook column.

Parameters **gradebook_column_id** (osid.id.Id) – a gradebook column Id

Returns the returned GradeEntry list

Return type osid.grading.GradeEntryList

Raise `NullArgument` – `gradebook_column_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Gradebook.`**`get_grade_entries_for_gradebook_column_on_date`**(*gradebook_column_id*, *from_*, *to*)

Gets a `GradeEntryList` for the given gradebook column and effective during the entire given date range inclusive but not confined to the date range.

Parameters

- **`gradebook_column_id`** (`osid.id.Id`) – a gradebook column `Id`
- **`from`** (`osid.calendaring.DateTime`) – start of date range
- **`to`** (`osid.calendaring.DateTime`) – end of date range

Returns the returned `GradeEntry` list

Return type `osid.grading.GradeEntryList`

Raise `InvalidArgument` – `from` is greater than `to`

Raise `NullArgument` – `gradebook_column_id, from, or to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Gradebook.`**`get_grade_entries_for_resource`**(*resource_id*)

Gets a `GradeEntryList` for the given key key resource.

Parameters **`resource_id`** (`osid.id.Id`) – a key resource `Id`

Returns the returned `GradeEntry` list

Return type `osid.grading.GradeEntryList`

Raise `NullArgument` – `resource_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Gradebook.`**`get_grade_entries_for_resource_on_date`**(*resource_id*, *from_*, *to*)

Gets a `GradeEntryList` for the given key resource and effective during the entire given date range inclusive but not confined to the date range.

Parameters

- **`resource_id`** (`osid.id.Id`) – a resource `Id`
- **`from`** (`osid.calendaring.DateTime`) – start of date range
- **`to`** (`osid.calendaring.DateTime`) – end of date range

Returns the returned `GradeEntry` list

Return type `osid.grading.GradeEntryList`

Raise `InvalidArgument` – `from` is greater than `to`

Raise `NullArgument` – `resource_id, from, or to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Gradebook.`**`get_grade_entries_for_gradebook_column_and_resource`**`(`*gradebook_column_id, resource_id*`)`

Gets a `GradeEntryList` for the gradebook column and key resource.

Parameters

- **`gradebook_column_id`** (`osid.id.Id`) – a gradebook column `Id`
- **`resource_id`** (`osid.id.Id`) – a key resource `Id`

Returns the returned `GradeEntry` list

Return type `osid.grading.GradeEntryList`

Raise `NullArgument` – `gradebook_column_id` or `resource_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Gradebook.`**`get_grade_entries_for_gradebook_column_and_resource_on_date`**`(`*gradebook_column_id, resource_id, from_, to*`)`

Gets a `GradeEntryList` for the given gradebook column, resource, and effective during the entire given date range inclusive but not confined to the date range.

Parameters

- **`gradebook_column_id`** (`osid.id.Id`) – a gradebook column `Id`
- **`resource_id`** (`osid.id.Id`) – a key resource `Id`
- **`from`** (`osid.calendaring.DateTime`) – start of date range
- **`to`** (`osid.calendaring.DateTime`) – end of date range

Returns the returned `GradeEntry` list

Return type `osid.grading.GradeEntryList`

Raise `InvalidArgument` – `from` is greater than `to`

Raise `NullArgument` – `gradebook_column_id, resource, from, or to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Gradebook.`**`get_grade_entries_by_grader`**`(`*resource_id*`)`

Gets a `GradeEntryList` for the given grader.

Parameters **`resource_id`** (`osid.id.Id`) – a resource `Id`

Returns the returned `GradeEntry` list

Return type `osid.grading.GradeEntryList`

Raise `NullArgument` – `resource_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Gradebook.`**`grade_entries`**

Gets all grade entries.

Returns a `GradeEntryList`

Return type `osid.grading.GradeEntryList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Grade Entry Query Methods

`Gradebook.`**`gradebook_id`**

Gets the `Gradebook Id` associated with this session.

Returns the `Gradebook Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`gradebook`**

Gets the `Gradebook` associated with this session.

Returns the `Gradebook` associated with this session

Return type `osid.grading.Gradebook`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Gradebook.`**`can_search_grade_entries`**`()`

Tests if this user can perform `GradeEntry` searches.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer search operations to unauthorized users.

Returns `false` if search methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`use_federated_gradebook_view`**`()`

Federates the view for methods in this session.

A federated view will include grade entries in gradebooks which are children of this gradebook in the gradebook hierarchy.

compliance: mandatory – This method is must be implemented.

`Gradebook.`**`use_isolated_gradebook_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts searches to this gradebook only.

compliance: mandatory – This method is must be implemented.

`Gradebook.`**`grade_entry_query`**

Gets a grade entry query.

Returns the grade entry query

Return type `osid.grading.GradeEntryQuery`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`get_grade_entries_by_query`**`(`*grade_entry_query*`)`

Gets a list of entries matching the given grade entry query.

Parameters **`grade_entry_query`** (`osid.grading.GradeEntryQuery`) – the grade entry query

Returns the returned `GradeEntryList`

Return type `osid.grading.GradeEntryList`

Raise `NullArgument` – `grade_entry_query` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `grade_entry_query` is not of this service

compliance: mandatory – This method must be implemented.

Grade Entry Admin Methods

`Gradebook.`**`gradebook_id`**

Gets the `Gradebook Id` associated with this session.

Returns the `Gradebook Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`gradebook`**

Gets the `Gradebook` associated with this session.

Returns the `Gradebook` associated with this session

Return type `osid.grading.Gradebook`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Gradebook.`**`can_create_grade_entries`**`()`

Tests if this user can create grade entries.

A return of true does not guarantee successful authorization. A return of false indicates that it is known creating a grade entry will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer create operations to an unauthorized user.

Returns `false` if `GradeEntry` creation is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`can_create_grade_entry_with_record_types`**(*grade_entry_record_types*)

Tests if this user can create a single `GradeEntry` using the desired record types.

While `GradingManager.getGradeEntryRecordTypes()` can be used to examine which records are supported, this method tests which record(s) are required for creating a specific `GradeEntry`. Providing an empty array tests if a `GradeEntry` can be created with no records.

Parameters **`grade_entry_record_types`** (`osid.type.Type[]`) – array of grade entry record types

Returns `true` if `GradeEntry` creation using the specified record `Types` is supported, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `grade_entry_record_types` is `null`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`get_grade_entry_form_for_create`**(*gradebook_column_id*, *resource_id*, *grade_entry_record_types*)

Gets the grade entry form for creating new grade entries.

A new form should be requested for each create transaction.

Parameters

- **`gradebook_column_id`** (`osid.id.Id`) – the gradebook column
- **`resource_id`** (`osid.id.Id`) – the key resource
- **`grade_entry_record_types`** (`osid.type.Type[]`) – array of grade entry record types

Returns the grade entry form

Return type `osid.grading.GradeEntryForm`

Raise `NotFound` – `gradebook_column_id or resource_id` not found

Raise `NullArgument` – `gradebook_column_id, resource_id,` or `grade_entry_record_types` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – unable to get form for requested record types

compliance: mandatory – This method must be implemented.

`Gradebook.`**`create_grade_entry`**(*grade_entry_form*)

Creates a new `GradeEntry`.

Parameters **`grade_entry_form`** (`osid.grading.GradeEntryForm`) – the form for this `GradeEntry`

Returns the new `GradeEntry`

Return type `osid.grading.GradeEntry`

Raise `IllegalState` – `grade_entry_form` already used in a create transaction

Raise `InvalidArgument` – one or more of the form elements is invalid

Raise `NullArgument` – `grade_entry_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `grade_entry_form` did not originate from `get_grade_entry_form_for_create()`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`can_overridecalculated_grade_entries`**`()`

Tests if this user can override grade entries calculated from another.

A return of true does not guarantee successful authorization. A return of false indicates that it is known creating a grade entry will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer create operations to an unauthorized user.

Returns `false` if `GradeEntry` override is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`get_grade_entry_form_for_override`**`(`*grade_entry_id*, *grade_entry_record_types*`)`

Gets the grade entry form for overriding calculated grade entries.

A new form should be requested for each create transaction.

Parameters

- **`grade_entry_id`** (`osid.id.Id`) – the `Id` of the grade entry to be overridden
- **`grade_entry_record_types`** (`osid.type.Type[]`) – array of grade entry record types

Returns the grade entry form

Return type `osid.grading.GradeEntryForm`

Raise `AlreadyExists` – `grade_entry_id` is already overridden

Raise `NotFound` – `grade_entry_id` not found or `grade_entry_id` is not a calculated entry

Raise `NullArgument` – `grade_entry_id` or `grade_entry_record_types` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – unable to get form for requested record types

compliance: mandatory – This method must be implemented.

`Gradebook.`**`override_calculated_grade_entry`**`(`*grade_entry_form*`)`

Creates a new overriding `GradeEntry`.

Parameters **`grade_entry_form`** (`osid.grading.GradeEntryForm`) – the form for this `GradeEntry`

Returns the new `GradeEntry`

Return type `osid.grading.GradeEntry`

Raise `IllegalState` – `grade_entry_form` already used in a create transaction

Raise `InvalidArgument` – one or more of the form elements is invalid

Raise `NullArgument` – `grade_entry_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `grade_entry_form` did not originate from `get_grade_entry_form_for_override()`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`can_update_grade_entries`**`()`

Tests if this user can update grade entries.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating a `GradeEntry` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer update operations to an unauthorized user.

Returns `false` if grade entry modification is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`get_grade_entry_form_for_update`**`(`*grade_entry_id*`)`

Gets the grade entry form for updating an existing entry.

A new grade entry form should be requested for each update transaction.

Parameters **`grade_entry_id`** (`osid.id.Id`) – the `Id` of the `GradeEntry`

Returns the grade entry form

Return type `osid.grading.GradeEntryForm`

Raise `NotFound` – `grade_entry_id` is not found

Raise `NullArgument` – `grade_entry_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Gradebook.`**`update_grade_entry`**`(`*grade_entry_form*`)`

Updates an existing grade entry.

Parameters **`grade_entry_form`** (`osid.grading.GradeEntryForm`) – the form containing the elements to be updated

Raise `IllegalState` – `grade_entry_form` already used in an update transaction

Raise `InvalidArgument` – the form contains an invalid value

Raise `NullArgument` – `grade_entry_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `grade_entry_form` did not originate from `get_grade_entry_form_for_update()`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`can_delete_grade_entries`**`()`

Tests if this user can delete grade entries.

A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting a `GradeEntry` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer delete operations to an unauthorized user.

Returns `false` if `GradeEntry` deletion is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`delete_grade_entry`**`(`*grade_entry_id*`)`

Deletes the `GradeEntry` identified by the given `Id`.

Parameters **`grade_entry_id`** (`osid.id.Id`) – the `Id` of the `GradeEntry` to delete

Raise `NotFound` – a `GradeEntry` was not found identified by the given `Id`

Raise `NullArgument` – `grade_entry_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Gradebook.`**`can_manage_grade_entry_aliases`**`()`

Tests if this user can manage `Id` aliases for `GradeEntries`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known changing an alias will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer alias operations to an unauthorized user.

Returns `false` if `GradeEntry` aliasing is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Gradebook.`**`alias_grade_entry`**`(`*grade_entry_id*, *alias_id*`)`

Adds an `Id` to a `GradeEntry` for the purpose of creating compatibility.

The primary `Id` of the `GradeEntry` is determined by the provider. The new `Id` performs as an alias to the primary `Id`. If the alias is a pointer to another grade entry, it is reassigned to the given grade entry `Id`.

Parameters

- **`grade_entry_id`** (`osid.id.Id`) – the `Id` of a `GradeEntry`
- **`alias_id`** (`osid.id.Id`) – the alias `Id`

Raise `AlreadyExists` – `alias_id` is already assigned

Raise `NotFound` – `grade_entry_id` not found

Raise `NullArgument` – `grade_entry_id` or `alias_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Objects

Grade

class `dlkit.grading.objects.`**`Grade`**

Bases: *`dlkit.osid.objects.OsidObject`*, *`dlkit.osid.markers.Subjugateable`*

A `Grade`.

Grades represent qualified performance levels defined within some grading system.

`grade_system_id`

Gets the `GradeSystem Id` in which this grade belongs.

Returns the grade system `Id`

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`grade_system`

Gets the `GradeSystem` in which this grade belongs.

Returns the grade system

Return type `osid.grading.GradeSystem`

compliance: mandatory – This method must be implemented.

`input_score_start_range`

Gets the low end of the input score range equivalent to this grade.

Returns the start range

Return type `decimal`

compliance: mandatory – This method must be implemented.

`input_score_end_range`

Gets the high end of the input score range equivalent to this grade.

Returns the end range

Return type `decimal`

compliance: mandatory – This method must be implemented.

`output_score`

Gets the output score for this grade used for calculating cumultives or performing articulation.

Returns the output score

Return type `decimal`

compliance: mandatory – This method must be implemented.

`get_grade_record`(*grade_record_type*)

Gets the grade record corresponding to the given `Grade` record `Type`.

This method is used to retrieve an object implementing the requested record. The `grade_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(grade_record_type)` is `true` .

Parameters **`grade_record_type`** (`osid.type.Type`) – the type of the record to retrieve

Returns the grade record

Return type `osid.grading.records.GradeRecord`

Raise `NullArgument` – `grade_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(grade_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Grade Form

class `dlkit.grading.objects.`**`GradeForm`**

Bases: *`dlkit.osid.objects.OsidObjectForm`*, *`dlkit.osid.objects.OsidSubjugateableForm`*

This is the form for creating and updating `Grades`.

Like all `OsidForm` objects, various data elements may be set here for use in the create and update methods in the `GradeAdminSession`. For each data element that may be set, metadata may be examined to provide display hints or data constraints.

`input_score_start_range_metadata`

Gets the metadata for the input score start range.

Returns metadata for the input score start range

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`input_score_start_range`

`input_score_end_range_metadata`

Gets the metadata for the input score start range.

Returns metadata for the input score start range

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`input_score_end_range`

`output_score_metadata`

Gets the metadata for the output score start range.

Returns metadata for the output score start range

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`output_score`

`get_grade_form_record`(*grade_record_type*)

Gets the `GradeFormRecord` corresponding to the given grade record `Type`.

Parameters **`grade_record_type`** (`osid.type.Type`) – the grade record type

Returns the grade form record

Return type `osid.grading.records.GradeFormRecord`

Raise `NullArgument` – `grade_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(grade_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Grade List

class `dlkit.grading.objects.`**`GradeList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists`, `GradeList` provides a means for accessing `Grade` elements sequentially either one at a time or many at a time.

Examples: while (gl.hasNext()) { Grade grade = gl.getNextGrade(); }

or

while (gl.hasNext()) { Grade[] grades = gl.getNextGrades(gl.available());

}

`next_grade`

Gets the next `Grade` in this list.

Returns the next `Grade` in this list. The `has_next()` method should be used to test that a next `Grade` is available before calling this method.

Return type `osid.grading.Grade`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_grades`(*n*)

Gets the next set of `Grade` elements in this list which must be less than or equal to the return from `available()`.

Parameters **`n`** (`cardinal`) – the number of `Grade` elements requested which must be less than or equal to `available()`

Returns an array of `Grade` elements.The length of the array is less than or equal to the number specified.

Return type `osid.grading.Grade`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Grade System

class `dlkit.grading.objects.`**`GradeSystem`**

Bases: *`dlkit.osid.objects.OsidObject`*, *`dlkit.osid.markers.Aggregateable`*

A `GradeSystem` represents a grading system.

The system can be based on assigned Grades or based on a numeric scale.

`is_based_on_grades``()`

Tests if the grading system is based on grades.

Returns true if the grading system is based on grades, `false` if the system is a numeric score

Return type `boolean`

compliance: mandatory – This method must be implemented.

`grade_ids`

Gets the grade `Ids` in this system ranked from highest to lowest.

Returns the list of grades `Ids`

Return type `osid.id.IdList`

Raise `IllegalState` – `is_based_on_grades()` is `false`

compliance: mandatory – This method must be implemented.

`grades`

Gets the grades in this system ranked from highest to lowest.

Returns the list of grades

Return type `osid.grading.GradeList`

Raise `IllegalState` – `is_based_on_grades()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`lowest_numeric_score`

Gets the lowest number in a numeric grading system.

Returns the lowest number

Return type `decimal`

Raise `IllegalState` – `is_based_on_grades()` is `true`

compliance: mandatory – This method must be implemented.

`numeric_score_increment`

Gets the incremental step.

Returns the increment

Return type `decimal`

Raise `IllegalState` – `is_based_on_grades()` is `true`

compliance: mandatory – This method must be implemented.

`highest_numeric_score`

Gets the highest number in a numeric grading system.

Returns the highest number

Return type `decimal`

Raise `IllegalState` – `is_based_on_grades()` is `true`

compliance: mandatory – This method must be implemented.

`get_grade_system_record`(*grade_system_record_type*)

Gets the grade system record corresponding to the given `GradeSystem` record `Type`.

This method is used to retrieve an object implementing the requested record. The `grade_system_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(grade_system_record_type)` is `true` .

Parameters **`grade_system_record_type`** (`osid.type.Type`) – the type of the record to retrieve

Returns the grade system record

Return type `osid.grading.records.GradeSystemRecord`

Raise `NullArgument` – `grade_system_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(grade_system_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Grade System Form

class `dlkit.grading.objects.`**`GradeSystemForm`**

Bases: *`dlkit.osid.objects.OsidObjectForm`*, *`dlkit.osid.objects.OsidAggregateableForm`*

This is the form for creating and updating `GradeSystems`.

Like all `OsidForm` objects, various data elements may be set here for use in the create and update methods in the `GradeSystemAdminSession`. For each data element that may be set, metadata may be examined to provide display hints or data constraints.

`based_on_grades_metadata`

Gets the metadata for a grade-based designation.

Returns metadata for the grade-based designation

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`based_on_grades`

`lowest_numeric_score_metadata`

Gets the metadata for the lowest numeric score.

Returns metadata for the lowest numeric score

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`lowest_numeric_score`

numeric_score_increment_metadata

Gets the metadata for the lowest numeric score.

Returns metadata for the lowest numeric score

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

numeric_score_increment

highest_numeric_score_metadata

Gets the metadata for the highest numeric score.

Returns metadata for the highest numeric score

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

highest_numeric_score

get_grade_system_form_record(*grade_system_record_type*)

Gets the `GradeSystemFormRecord` corresponding to the given grade system record `Type`.

Parameters **grade_system_record_type** (`osid.type.Type`) – the grade system record type

Returns the grade system form record

Return type `osid.grading.records.GradeSystemFormRecord`

Raise `NullArgument` – `grade_system_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(grade_system_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Grade System List

class `dlkit.grading.objects.`**GradeSystemList**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists`, `GradeSystemList` provides a means for accessing `GradeSystem` elements sequentially either one at a time or many at a time.

Examples: while (gsl.hasNext()) { GradeSystem system = gsl.getNextGradeSystem(); }

or

while (gsl.hasNext()) { GradeSystem[] systems = gsl.getNextGradeSystems(gsl.available());

}

next_grade_system

Gets the next `GradeSystem` in this list.

Returns the next `GradeSystem` in this list. The `has_next()` method should be used to test that a next `GradeSystem` is available before calling this method.

Return type `osid.grading.GradeSystem`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

get_next_grade_systems(*n*)

Gets the next set of `GradeSystem` elements in this list which must be less than or equal to the return from `available()`.

Parameters **n** (`cardinal`) – the number of `GradeSystem` elements requested which must be less than or equal to `available()`

Returns an array of `GradeSystem` elements.The length of the array is less than or equal to the number specified.

Return type `osid.grading.GradeSystem`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Grade Entry

class `dlkit.grading.objects.`**GradeEntry**

Bases: *`dlkit.osid.objects.OsidRelationship`*

A `GradeEntry` represents an entry in a `Gradebook`.

gradebook_column_id

Gets the `Id` of the `GradebookColumn`.

Returns the `Id` of the `GradebookColumn`

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

gradebook_column

Gets the `GradebookColumn`.

Returns the `GradebookColumn`

Return type `osid.grading.GradebookColumn`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

key_resource_id

Gets the `Id` of the key resource of this entry.

The key resource may be a student or other applicable key to identify a row of grading entries.

Returns `Id` of the key resource

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

key_resource

Gets the key resource of this entry.

The key resource may be a student or other applicable key to identify a row of grading entries.

Returns the key resource

Return type `osid.resource.Resource`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

is_derived()

Tests if this is a calculated entry.

Returns `true` if this entry is a calculated entry, `false` otherwise. If `true`, then `overrides_calculated_entry()` must be `false`.

Return type `boolean`

compliance: mandatory – This method must be implemented.

overrides_calculated_entry()

Tests if this is a manual entry that overrides a calculated entry.

Returns `true` if this entry overrides a calculated entry, `false` otherwise. If `true`, then `is_derived()` must be `false`.

Return type `boolean`

compliance: mandatory – This method must be implemented.

overridden_calculated_entry_id

Gets the calculated entry `Id` this entry overrides.

Returns the calculated entry `Id`

Return type `osid.id.Id`

Raise `IllegalState` – `overrides_derived_entry()` is `false`

compliance: mandatory – This method must be implemented.

overridden_calculated_entry

Gets the calculated entry this entry overrides.

Returns the calculated entry

Return type `osid.grading.GradeEntry`

Raise `IllegalState` – `overrides_calculated_entry()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

is_ignored_for_calculations()

Tests if this is entry should be ignored in any averaging, scaling or curve calculation.

Returns `true` if this entry is ignored, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

is_graded()

Tests if a grade or score has been assigned to this entry.

Generally, an entry is created with a grade or score.

Returns `true` if a grade has been assigned, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

grade_id

Gets the grade `Id` in this entry if the grading system is based on grades.

Returns the grade `Id`

Return type `osid.id.Id`

Raise `IllegalState` – `is_graded()` is `false` or `GradeSystem.isBasedOnGrades()` is `false`

compliance: mandatory – This method must be implemented.

grade

Gets the grade in this entry if the grading system is based on grades.

Returns the grade

Return type `osid.grading.Grade`

Raise `IllegalState` – `is_graded()` is `false` or `GradeSystem.isBasedOnGrades()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

score

Gets the score in this entry if the grading system is not based on grades.

Returns the score

Return type `decimal`

Raise `IllegalState` – `is_graded()` is `false` or `GradeSystem.isBasedOnGrades()` is `true`

compliance: mandatory – This method must be implemented.

time_graded

Gets the time the gradeable object was graded.

Returns the timestamp of the grading entry

Return type `osid.calendaring.DateTime`

Raise `IllegalState` – `is_graded()` is `false` or `is_derived()` is `true`

compliance: mandatory – This method must be implemented.

grader_id

Gets the `Id` of the `Resource` that created this entry.

Returns the `Id` of the `Resource`

Return type `osid.id.Id`

Raise `IllegalState` – `is_graded()` is `false` or `is_derived()` is `true`

compliance: mandatory – This method must be implemented.

grader

Gets the `Resource` that created this entry.

Returns the `Resource`

Return type `osid.resource.Resource`

Raise `IllegalState` – `is_graded() is false or is_derived() is true`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`grading_agent_id`

Gets the `Id` of the `Agent` that created this entry.

Returns the `Id` of the `Agent`

Return type `osid.id.Id`

Raise `IllegalState` – `is_graded()` is `false` or `is_derived()` is `true`

compliance: mandatory – This method must be implemented.

`grading_agent`

Gets the `Agent` that created this entry.

Returns the `Agent`

Return type `osid.authentication.Agent`

Raise `IllegalState` – `is_graded() is false or is_derived() is true`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_grade_entry_record`(*grade_entry_record_type*)

Gets the grade entry record corresponding to the given `GradeEntry` record `Type`.

This method is used to retrieve an object implementing the requested record. The `grade_entry_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(grade_entry_record_type)` is `true`.

Parameters **`grade_entry_record_type`** (`osid.type.Type`) – the type of the record to retrieve

Returns the grade entry record

Return type `osid.grading.records.GradeEntryRecord`

Raise `NullArgument` – `grade_entry_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(grade_entry_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Grade Entry Form

class `dlkit.grading.objects.`**`GradeEntryForm`**

Bases: *`dlkit.osid.objects.OsidRelationshipForm`*

This is the form for creating and updating `GradeEntries`.

Like all `OsidForm` objects, various data elements may be set here for use in the create and update methods in the `GradeEntryAdminSession`. For each data element that may be set, metadata may be examined to provide display hints or data constraints.

`ignored_for_calculations_metadata`

Gets the metadata for the ignore flag.

Returns metadata for the ignore flag

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`ignored_for_calculations`

`grade_metadata`
Gets the metadata for a grade.

Returns metadata for the grade

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`grade`

`score_metadata`
Gets the metadata for a score.

Returns metadata for the score

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`score`

`get_grade_entry_form_record`(*grade_entry_record_type*)
Gets the `GradeEntryFormRecord` corresponding to the given grade entry record `Type`.

Parameters **`grade_entry_record_type`** (`osid.type.Type`) – the grade entry record type

Returns the grade entry form record

Return type `osid.grading.records.GradeEntryFormRecord`

Raise `NullArgument` – `grade_entry_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(grade_entry_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Grade Entry List

class `dlkit.grading.objects.`**`GradeEntryList`**
Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists, GradeEntryList` provides a means for accessing `GradeEntry` elements sequentially either one at a time or many at a time.

Examples: while (gel.hasNext()) { GradeEntry entry = gel.getNextGradeEntry(); }

or

while (gel.hasNext()) { GradeEntry[] entries = gel.getNextGradeEntries(gel.available());

}

`next_grade_entry`
Gets the next `GradeEntry` in this list.

Returns the next `GradeEntry` in this list. The `has_next()` method should be used to test that a next `GradeEntry` is available before calling this method.

Return type `osid.grading.GradeEntry`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

get_next_grade_entries(*n*)

Gets the next set of `GradeEntry` elements in this list which must be less than or equal to the number returned from `available()`.

Parameters n (`cardinal`) – the number of `GradeEntry` elements requested which should be less than or equal to `available()`

Returns an array of `GradeEntry` elements.The length of the array is less than or equal to the number specified.

Return type `osid.grading.GradeEntry`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Gradebook Column

class `dlkit.grading.objects.`**GradebookColumn**

Bases: *`dlkit.osid.objects.OsidObject`*

A `GradebookColumn` represents a series of grade entries in a gradebook.

Each GradeEntry in a column share the same `GradeSystem`.

grade_system_id

Gets the `GradeSystem Id` in which this grade belongs.

Returns the grade system `Id`

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

grade_system

Gets the `GradeSystem` in which this grade belongs.

Returns the package grade system

Return type `osid.grading.GradeSystem`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

get_gradebook_column_record(*gradebook_column_record_type*)

Gets the gradebook column record corresponding to the given `GradeBookColumn` record `Type`.

This method ie used to retrieve an object implementing the requested record. The `gradebook_column_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(gradebook_column_record_type)` is `true` .

Parameters gradebook_column_record_type (`osid.type.Type`) – the type of the record to retrieve

Returns the gradebook column record

Return type `osid.grading.records.GradebookColumnRecord`

Raise `NullArgument` – `gradebook_column_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(gradebook_column_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Gradebook Column Form

class `dlkit.grading.objects.`**`GradebookColumnForm`**

Bases: *`dlkit.osid.objects.OsidObjectForm`*

This is the form for creating and updating `GradebookColumns`.

Like all `OsidForm` objects, various data elements may be set here for use in the create and update methods in the `GradebookAdminSession`. For each data element that may be set, metadata may be examined to provide display hints or data constraints.

`grade_system_metadata`

Gets the metadata for a grade system.

Returns metadata for the grade system

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`grade_system`

`get_gradebook_column_form_record`(*gradebook_column_record_type*)

Gets the `GradebookColumnFormRecord` corresponding to the given gradebook column record `Type`.

Parameters **`gradebook_column_record_type`** (`osid.type.Type`) – a gradebook column record type

Returns the gradebook column form record

Return type `osid.grading.records.GradebookColumnFormRecord`

Raise `NullArgument` – `gradebook_column_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(gradebook_column_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Gradebook Column List

class `dlkit.grading.objects.`**`GradebookColumnList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists, GradebookColumnList` provides a means for accessing `GradebookColumn` elements sequentially either one at a time or many at a time.

Examples: while (gcl.hasNext()) { GradebookColumn column = gcl.getNextGradebookColumn(); }

or

while (gcl.hasNext()) { GradebookColumn[] columns = gcl.getNextGradebookColumns(gcl.available());

}

`next_gradebook_column`
Gets the next `GradebookColumn` in this list.

Returns the next `GradebookColumn` in this list. The `has_next()` method should be used to test that a next `GradebookColumn` is available before calling this method.

Return type `osid.grading.GradebookColumn`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_gradebook_columns`(*n*)
Gets the next set of `GradebookColumn` elements in this list which must be less than or equal to the return from `available()`.

Parameters **n** (`cardinal`) – the number of `GradebookColumn` elements requested which must be less than or equal to `available()`

Returns an array of `GradebookColumn` elements.The length of the array is less than or equal to the number specified.

Return type `osid.grading.GradebookColumn`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Gradebook Column Summary

class `dlkit.grading.objects.`**`GradebookColumnSummary`**
Bases: *`dlkit.osid.objects.OsidObject`*

A `GradebookColumnSummary` is a summary of all entries within a gradebook column.

`gradebook_column_id`
Gets the `Id` of the `GradebookColumn`.

Returns the `Id` of the `GradebookColumn`

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`gradebook_column`
Gets the `GradebookColumn`.

Returns the `GradebookColumn`

Return type `osid.grading.GradebookColumn`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`mean`

Gets the mean score.

If this system is based on grades, the mean output score is returned.

Returns the mean score

Return type `decimal`

compliance: mandatory – This method must be implemented.

`median`

Gets the median score.

If this system is based on grades, the mean output score is returned.

Returns the median score

Return type `decimal`

compliance: mandatory – This method must be implemented.

`mode`

Gets the mode of the score.

If this system is based on grades, the mode of the output score is returned.

Returns the median score

Return type `decimal`

compliance: mandatory – This method must be implemented.

`rms`

Gets the root mean square of the score.

If this system is based on grades, the RMS of the output score is returned.

Returns the median score

Return type `decimal`

compliance: mandatory – This method must be implemented.

`standard_deviation`

Gets the standard deviation.

If this system is based on grades, the spread of the output scores is returned.

Returns the standard deviation

Return type `decimal`

compliance: mandatory – This method must be implemented.

`sum`

Gets the sum of the scores.

If this system is based on grades, the sum of the output scores is returned.

Returns the median score

Return type `decimal`

compliance: mandatory – This method must be implemented.

get_gradebook_column_summary_record(*gradebook_column_summary_record_type*)

Gets the gradebook column summary record corresponding to the given `GradebookColumnSummary` record `Type`.

This method is used to retrieve an object implementing the requested record. The `gradebook_column_summary_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(gradebook_column_summary_record_type)` is `true` .

Parameters **gradebook_column_summary_record_type** (`osid.type.Type`) – the type of the record to retrieve

Returns the gradebook column summary record

Return type `osid.grading.records.GradebookColumnSummaryRecord`

Raise `NullArgument` – `gradebook_column_summary_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(gradebook_column_summary_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Gradebook

class `dlkit.grading.objects.`**Gradebook**(*abc_grading_objects.Gradebook, osid_objects.OsidCatalog*)

:noindex:

get_gradebook_record(*gradebook_record_type*)

Gets the gradebook record corresponding to the given `Gradebook` record `Type`.

This method is used to retrieve an object implementing the requested record. The `gradebook_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(gradebook_record_type)` is `true` .

Parameters **gradebook_record_type** (`osid.type.Type`) – a gradebook record type

Returns the gradebook record

Return type `osid.grading.records.GradebookRecord`

Raise `NullArgument` – `gradebook_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(gradebook_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Gradebook Form

class `dlkit.grading.objects.`**GradebookForm**

Bases: *`dlkit.osid.objects.OsidCatalogForm`*

This is the form for creating and updating `Gradebooks`.

Like all `OsidForm` objects, various data elements may be set here for use in the create and update methods in the `GradebookAdminSession`. For each data element that may be set, metadata may be examined to provide display hints or data constraints.

`get_gradebook_form_record`(*gradebook_record_type*)

Gets the `GradebookFormRecord` corresponding to the given gradebook record `Type`.

Parameters **`gradebook_record_type`** (`osid.type.Type`) – a gradebook record type

Returns the gradebook form record

Return type `osid.grading.records.GradebookFormRecord`

Raise `NullArgument` – `gradebook_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(gradebook_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Gradebook List

class `dlkit.grading.objects.`**`GradebookList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists`, `GradebookList` provides a means for accessing `Gradebook` elements sequentially either one at a time or many at a time.

Examples: while (gl.hasNext()) { Gradebook gradebook = gl.getNextGradebook(); }

or

while (gl.hasNext()) { Gradebook[] gradebooks = gl.getNextGradebooks(gl.available());

}

`next_gradebook`

Gets the next `Gradebook` in this list.

Returns the next `Gradebook` in this list. The `has_next()` method should be used to test that a next `Gradebook` is available before calling this method.

Return type `osid.grading.Gradebook`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_gradebooks`(*n*)

Gets the next set of `Gradebook` elements in this list which must be less than or equal to the return from `available()`.

Parameters **`n`** (`cardinal`) – the number of `Gradebook` elements requested which must be less than or equal to `available()`

Returns an array of `Gradebook` elements.The length of the array is less than or equal to the number specified.

Return type `osid.grading.Gradebook`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Gradebook Node

class `dlkit.grading.objects.`**`GradebookNode`**

Bases: `dlkit.osid.objects.OsidNode`

This interface is a container for a partial hierarchy retrieval.

The number of hierarchy levels traversable through this interface depend on the number of levels requested in the `GradebookHierarchySession`.

`gradebook`

Gets the `Gradebook` at this node.

Returns the gradebook represented by this node

Return type `osid.grading.Gradebook`

compliance: mandatory – This method must be implemented.

`parent_gradebook_nodes`

Gets the parents of this gradebook.

Returns the parents of the `id`

Return type `osid.grading.GradebookNodeList`

compliance: mandatory – This method must be implemented.

`child_gradebook_nodes`

Gets the children of this gradebook.

Returns the children of this gradebook

Return type `osid.grading.GradebookNodeList`

compliance: mandatory – This method must be implemented.

Gradebook Node List

class `dlkit.grading.objects.`**`GradebookNodeList`**

Bases: `dlkit.osid.objects.OsidList`

Like all `OsidLists, GradebookNodeList` provides a means for accessing `GradebookNode` elements sequentially either one at a time or many at a time.

Examples: while (gnl.hasNext()) { GradebookNode node = gnl.getNextGradebookNode(); }

or

while (gnl.hasNext()) { GradebookNode[] nodes = gnl.getNextGradebookNodes(gnl.available());

}

`next_gradebook_node`

Gets the next `GradebookNode` in this list.

Returns the next `GradebookNode` in this list. The `has_next()` method should be used to test that a next `GradebookNode` is available before calling this method.

Return type `osid.grading.GradebookNode`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

get_next_gradebook_nodes(*n*)

Gets the next set of `GradebookNode` elements in this list which must be less than or equal to the return from `available()`.

Parameters **n** (`cardinal`) – the number of `GradebookNode` elements requested which must be less than or equal to `available()`

Returns an array of `GradebookNode` elements.The length of the array is less than or equal to the number specified.

Return type `osid.grading.GradebookNode`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Queries

Grade Query

class `dlkit.grading.queries.`**GradeQuery**

Bases: *dlkit.osid.queries.OsidObjectQuery*, *dlkit.osid.queries.OsidSubjugateableQuery*

This is the query for searching gradings.

Each method match request produces an `AND` term while multiple invocations of a method produces a nested `OR`.

match_grade_system_id(*grade_system_id*, *match*)

Sets the grade system `Id` for this query.

Parameters

- **grade_system_id** (`osid.id.Id`) – a grade system `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for negative match

Raise `NullArgument` – `grade_system_id` is `null`

compliance: mandatory – This method must be implemented.

grade_system_id_terms

supports_grade_system_query()

Tests if a `GradeSystemQuery` is available for querying grade systems.

Returns `true` if a grade system query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

grade_system_query

Gets the query for a grade system.

Multiple retrievals produce a nested `OR` term.

Returns the grade system query

Return type `osid.grading.GradeSystemQuery`

Raise `Unimplemented` – `supports_grade_system_query()` is `false`

compliance: optional – This method must be implemented if ``supports_grade_system_query()`` is ``true``.

`grade_system_terms`

`match_input_score_start_range`(*start*, *end*, *match*)
Matches grades with the start input score inclusive.

Parameters

- **`start`** (`decimal`) – start of range
- **`end`** (`decimal`) – end of range
- **`match`** (`boolean`) – `true` for a positive match, `false` for negative match

Raise `InvalidArgument` – `start` is greater than `end`

compliance: mandatory – This method must be implemented.

`input_score_start_range_terms`

`match_input_score_end_range`(*start*, *end*, *match*)
Matches grades with the end input score inclusive.

Parameters

- **`start`** (`decimal`) – start of range
- **`end`** (`decimal`) – end of range
- **`match`** (`boolean`) – `true` for a positive match, `false` for negative match

Raise `InvalidArgument` – `start` is greater than `end`

compliance: mandatory – This method must be implemented.

`input_score_end_range_terms`

`match_input_score`(*start*, *end*, *match*)
Matches grades with the input score range contained within the given range inclusive.

Parameters

- **`start`** (`decimal`) – start of range
- **`end`** (`decimal`) – end of range
- **`match`** (`boolean`) – `true` for a positive match, `false` for negative match

Raise `InvalidArgument` – `start` is greater than `end`

compliance: mandatory – This method must be implemented.

`input_score_terms`

`match_output_score`(*start*, *end*, *match*)
Matches grades with the output score contained within the given range inclusive.

Parameters

- **`start`** (`decimal`) – start of range
- **`end`** (`decimal`) – end of range
- **`match`** (`boolean`) – `true` for a positive match, `false` for negative match

Raise `InvalidArgument` – `start` is greater than `end`

compliance: mandatory – This method must be implemented.

`output_score_terms`

`match_grade_entry_id`(*grade_entry_id*, *match*)
Sets the grade entry `Id` for this query.

Parameters

- **`grade_entry_id`** (`osid.id.Id`) – a grade entry `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for negative match

Raise `NullArgument` – `grade_entry_id` is `null`

compliance: mandatory – This method must be implemented.

`grade_entry_id_terms`

`supports_grade_entry_query`()
Tests if a `GradeEntryQuery` is available for querying grade entries.

Returns `true` if a grade entry query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`grade_entry_query`
Gets the query for a grade entry.

Multiple retrievals produce a nested `OR` term.

Returns the grade entry query

Return type `osid.grading.GradeEntryQuery`

Raise `Unimplemented` – `supports_grade_entry_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_grade_entry_query()‘‘ is ‘‘true‘‘.

`match_any_grade_entry`(*match*)
Matches grades that are assigned to any grade entry.

Parameters **`match`** (`boolean`) – `true` to match grades used in any grade entry, `false` to match grades that are not used in any grade entries

compliance: mandatory – This method must be implemented.

`grade_entry_terms`

`match_gradebook_id`(*gradebook_id*, *match*)
Sets the gradebook `Id` for this query.

Parameters

- **`gradebook_id`** (`osid.id.Id`) – a gradebook `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for negative match

Raise `NullArgument` – `gradebook_id` is `null`

compliance: mandatory – This method must be implemented.

`gradebook_id_terms`

supports_gradebook_query()

Tests if a GradebookQuery is available.

Returns true if a gradebook query is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

gradebook_query

Gets the query for a gradebook.

Multiple retrievals produce a nested OR term.

Returns the gradebook query

Return type osid.grading.GradebookQuery

Raise Unimplemented – supports_gradebook_query() is false

compliance: optional – This method must be implemented if ‘‘supports_gradebook_column_query()‘‘ is ‘‘true‘‘.

gradebook_terms

get_grade_query_record(*grade_record_type*)

Gets the grade query record corresponding to the given Grade record Type.

Multiple retrievals produce a nested OR term.

Parameters **grade_record_type** (osid.type.Type) – a grade record type

Returns the grade query record

Return type osid.grading.records.GradeQueryRecord

Raise NullArgument – grade_record_type is null

Raise OperationFailed – unable to complete request

Raise Unsupported – has_record_type(grade_record_type) is false

compliance: mandatory – This method must be implemented.

Grade System Query

class dlkit.grading.queries.**GradeSystemQuery**

Bases: *dlkit.osid.queries.OsidObjectQuery*, *dlkit.osid.queries.OsidAggregateableQuery*

This is the query for searching grade systems.

Each method match request produces an AND term while multiple invocations of a method produces a nested OR.

match_based_on_grades(*match*)

Matches grade systems based on grades.

Parameters **match** (boolean) – true for a positive match, false for negative match

compliance: mandatory – This method must be implemented.

based_on_grades_terms

match_grade_id(*grade_id*, *match*)

Sets the grade Id for this query.

Parameters

- **grade_id** (`osid.id.Id`) – a grade `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for negative match

Raise `NullArgument` – `grade_id` is `null`

compliance: mandatory – This method must be implemented.

grade_id_terms

supports_grade_query()

Tests if a `GradeQuery` is available for querying grades.

Returns `true` if a grade query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

grade_query

Gets the query for a grade.

Multiple retrievals produce a nested `OR` term.

Returns the grade query

Return type `osid.grading.GradeQuery`

Raise `Unimplemented` – `supports_grade_query()` is `false`

compliance: optional – This method must be implemented if ``supports_grade_query()`` is ``true``.

match_any_grade(*match*)

Matches grade systems with any grade.

Parameters match (`boolean`) – `true` to match grade systems with any grade, `false` to match systems with no grade

compliance: mandatory – This method must be implemented.

grade_terms

match_lowest_numeric_score(*start*, *end*, *match*)

Matches grade systems whose low end score falls in the specified range inclusive.

Parameters

- **start** (`decimal`) – low end of range
- **end** (`decimal`) – high end of range
- **match** (`boolean`) – `true` for a positive match, `false` for negative match

Raise `InvalidArgument` – `end` is less than `start`

Raise `NullArgument` – `grade_id` is `null`

compliance: mandatory – This method must be implemented.

lowest_numeric_score_terms

match_numeric_score_increment(*start*, *end*, *match*)

Matches grade systems numeric score increment is between the specified range inclusive.

Parameters

- **start** (`decimal`) – low end of range

- **end** (`decimal`) – high end of range
- **match** (`boolean`) – `true` for a positive match, `false` for negative match

Raise `InvalidArgument` – `end` is less than `start`

Raise `NullArgument` – `grade_id` is `null`

compliance: mandatory – This method must be implemented.

`numeric_score_increment_terms`

`match_highest_numeric_score`(*start*, *end*, *match*)

Matches grade systems whose high end score falls in the specified range inclusive.

Parameters

- **start** (`decimal`) – low end of range
- **end** (`decimal`) – high end of range
- **match** (`boolean`) – `true` for a positive match, `false` for negative match

Raise `InvalidArgument` – `end` is less than `start`

Raise `NullArgument` – `grade_id` is `null`

compliance: mandatory – This method must be implemented.

`highest_numeric_score_terms`

`match_gradebook_column_id`(*gradebook_column_id*, *match*)

Sets the gradebook column `Id` for this query.

Parameters

- **gradebook_column_id** (`osid.id.Id`) – a gradebook column `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for negative match

Raise `NullArgument` – `gradebook_column_id` is `null`

compliance: mandatory – This method must be implemented.

`gradebook_column_id_terms`

`supports_gradebook_column_query`()

Tests if a `GradebookColumnQuery` is available.

Returns `true` if a gradebook column query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`gradebook_column_query`

Gets the query for a gradebook column.

Multiple retrievals produce a nested `OR` term.

Returns the gradebook column query

Return type `osid.grading.GradebookColumnQuery`

Raise `Unimplemented` – `supports_gradebook_column_query()` is `false`

compliance: optional – This method must be implemented if ``supports_gradebook_column_query()`` is ``true``.

match_any_gradebook_column(*match*)
Matches grade systems assigned to any gradebook column.

Parameters match (`boolean`) – `true` to match grade systems mapped to any column, `false` to match systems mapped to no columns

compliance: mandatory – This method must be implemented.

gradebook_column_terms

match_gradebook_id(*gradebook_id*, *match*)
Sets the gradebook `Id` for this query.

Parameters

- **gradebook_id** (`osid.id.Id`) – a gradebook `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for negative match

Raise `NullArgument` – `gradebook_id` is `null`

compliance: mandatory – This method must be implemented.

gradebook_id_terms

supports_gradebook_query()
Tests if a `GradebookQuery` is available.

Returns `true` if a gradebook query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

gradebook_query
Gets the query for a gradebook.

Multiple retrievals produce a nested `OR` term.

Returns the gradebook query

Return type `osid.grading.GradebookQuery`

Raise `Unimplemented` – `supports_gradebook_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_gradebook_query()‘‘ is ‘‘true‘‘.

gradebook_terms

get_grade_system_query_record(*grade_system_record_type*)
Gets the grade system query record corresponding to the given `GradeSystem` record `Type`.

Multiple retrievals produce a nested `OR` term.

Parameters grade_system_record_type (`osid.type.Type`) – a grade system record type

Returns the grade system query record

Return type `osid.grading.records.GradeSystemQueryRecord`

Raise `NullArgument` – `grade_system_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(grade_system_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Grade Entry Query

class `dlkit.grading.queries.`**`GradeEntryQuery`**

Bases: *`dlkit.osid.queries.OsidRelationshipQuery`*

This is the query for searching grade entries.

Each method match request produces an `AND` term while multiple invocations of a method produces a nested `OR`.

`match_gradebook_column_id`(*gradebook_column_id*, *match*)

Sets the gradebook column `Id` for this query.

Parameters

- **`gradebook_column_id`** (`osid.id.Id`) – a gradebook column `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `gradebook_column_id` is `null`

compliance: mandatory – This method must be implemented.

`gradebook_column_id_terms`

`supports_gradebook_column_query`()

Tests if a `GradebookColumnQuery` is available for querying creators.

Returns `true` if a gradebook column query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`gradebook_column_query`

Gets the query for a gradebook column.

Multiple retrievals produce a nested `OR` term.

Returns the gradebook column query

Return type `osid.grading.GradebookColumnQuery`

Raise `Unimplemented` – `supports_gradebook_column_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_gradebook_column_query()‘‘ is ‘‘true‘‘.

`gradebook_column_terms`

`match_key_resource_id`(*resource_id*, *match*)

Sets the key resource `Id` for this query.

Parameters

- **`resource_id`** (`osid.id.Id`) – a resource `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `resource_id` is `null`

compliance: mandatory – This method must be implemented.

`key_resource_id_terms`

`supports_key_resource_query`()

Tests if a `ResourceQUery` is available for querying key resources.

Returns `true` if a resource query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`key_resource_query`

Gets the query for a key resource.

Multiple retrievals produce a nested `OR` term.

Returns the resource query

Return type `osid.resource.ResourceQuery`

Raise `Unimplemented` – `supports_key_resource_query()` is `false`

compliance: optional – This method must be implemented if ``supports_key_resource_query()`` is ``true``.

`match_any_key_resource`(*match*)

Matches grade entries with any key resource.

Parameters **`match`** (`boolean`) – `true` to match grade entries with any key resource, `false` to match entries with no key resource

compliance: mandatory – This method must be implemented.

`key_resource_terms`

`match_derived`(*match*)

Matches derived grade entries.

Parameters **`match`** (`boolean`) – `true` to match derived grade entries , `false` to match manual entries

compliance: mandatory – This method must be implemented.

`derived_terms`

`match_overridden_grade_entry_id`(*grade_entry_id*, *match*)

Sets the grade entry `Id` for an overridden calculated grade entry.

Parameters

- **`grade_entry_id`** (`osid.id.Id`) – a grade entry `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `grade_entry_id` is `null`

compliance: mandatory – This method must be implemented.

`overridden_grade_entry_id_terms`

`supports_overridden_grade_entry_query`()

Tests if a `GradeEntry` is available for querying overridden calculated grade entries.

Returns `true` if a grade entry query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`overridden_grade_entry_query`

Gets the query for an overridden derived grade entry.

Multiple retrievals produce a nested `OR` term.

Returns the grade entry query

Return type `osid.grading.GradeEntryQuery`

Raise `Unimplemented` – `supports_overridden_grade_entry_query()` is `false`

compliance: optional – This method must be implemented if ``supports_overridden_grade_entry_query()`` is ``true``.

match_any_overridden_grade_entry(*match*)
Matches grade entries overriding any calculated grade entry.

Parameters **match** (`boolean`) – `true` to match grade entries overriding any grade entry, `false` to match entries not overriding any entry

compliance: mandatory – This method must be implemented.

overridden_grade_entry_terms

match_ignored_for_calculations(*match*)
Matches grade entries ignored for calculations.

Parameters **match** (`boolean`) – `true` to match grade entries ignored for calculations, `false` to match entries used in calculations

compliance: mandatory – This method must be implemented.

ignored_for_calculations_terms

match_grade_id(*grade_id*, *match*)
Sets the grade `Id` for this query.

Parameters

- **grade_id** (`osid.id.Id`) – a grade `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `grade_id` is `null`

compliance: mandatory – This method must be implemented.

grade_id_terms

supports_grade_query()
Tests if a `GradeQuery` is available for querying grades.

Returns `true` if a grade query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

grade_query
Gets the query for a grade.

Multiple retrievals produce a nested `OR` term.

Returns the grade query

Return type `osid.grading.GradeQuery`

Raise `Unimplemented` – `supports_grade_query()` is `false`

compliance: optional – This method must be implemented if ``supports_grade_query()`` is ``true``.

match_any_grade(*match*)
Matches grade entries with any grade.

Parameters **match** (`boolean`) – `true` to match grade entries with any grade, `false` to match entries with no grade

compliance: mandatory – This method must be implemented.

grade_terms

match_score (*start*, *end*, *match*)
Matches grade entries which score is between the specified score inclusive.

Parameters

- **start** (`decimal`) – start of range
- **end** (`decimal`) – end of range
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `end` is less than `start`

compliance: mandatory – This method must be implemented.

match_any_score (*match*)
Matches grade entries with any score.

Parameters **match** (`boolean`) – `true` to match grade entries with any score, `false` to match entries with no score

compliance: mandatory – This method must be implemented.

score_terms

match_time_graded (*start*, *end*, *match*)
Matches grade entries which graded time is between the specified times inclusive.

Parameters

- **start** (`osid.calendaring.DateTime`) – start of range
- **end** (`osid.calendaring.DateTime`) – end of range
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `end` is less than `start`

compliance: mandatory – This method must be implemented.

time_graded_terms

match_grader_id (*resource_id*, *match*)
Sets the agent `Id` for this query.

Parameters

- **resource_id** (`osid.id.Id`) – a resource `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `resource_id` is `null`

compliance: mandatory – This method must be implemented.

grader_id_terms

supports_grader_query ()
Tests if a `ResourceQuery` is available for querying graders.

Returns `true` if a resource query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`grader_query`

Gets the query for an agent.

Multiple retrievals produce a nested `OR` term.

Returns the resource query

Return type `osid.resource.ResourceQuery`

Raise `Unimplemented` – `supports_resource_query()` is `false`

compliance: optional – This method must be implemented if ``supports_resource_query()`` is ``true``.

`match_any_grader`(*match*)

Matches grade entries with any grader.

Parameters **`match`** (`boolean`) – `true` to match grade entries with any grader, `false` to match entries with no grader

compliance: mandatory – This method must be implemented.

`grader_terms`

`match_grading_agent_id`(*agent_id*, *match*)

Sets the grading agent `Id` for this query.

Parameters

- **`agent_id`** (`osid.id.Id`) – an agent `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `agent_id` is `null`

compliance: mandatory – This method must be implemented.

`grading_agent_id_terms`

`supports_grading_agent_query`()

Tests if an `AgentQuery` is available for querying grading agents.

Returns `true` if an agent query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`grading_agent_query`

Gets the query for an agent.

Multiple retrievals produce a nested `OR` term.

Returns the agent query

Return type `osid.authentication.AgentQuery`

Raise `Unimplemented` – `supports_grading_agent_query()` is `false`

compliance: optional – This method must be implemented if ``supports_grading_agent_query()`` is ``true``.

`match_any_grading_agent`(*match*)

Matches grade entries with any grading agent.

Parameters **match** (boolean) – true to match grade entries with any grading agent, false to match entries with no grading agent

compliance: mandatory – This method must be implemented.

grading_agent_terms

match_gradebook_id(*gradebook_id*, *match*)
Sets the gradebook Id for this query.

Parameters

- **gradebook_id** (osid.id.Id) – a gradebook Id
- **match** (boolean) – true for a positive match, false for a negative match

Raise NullArgument – gradebook_id is null

compliance: mandatory – This method must be implemented.

gradebook_id_terms

supports_gradebook_query()
Tests if a GradebookQuery is available for querying resources.

Returns true if a gradebook query is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

gradebook_query
Gets the query for a gradebook.

Multiple retrievals produce a nested OR term.

Returns the gradebook query

Return type osid.grading.GradebookQuery

Raise Unimplemented – supports_gradebook_query() is false

compliance: optional – This method must be implemented if ``supports_gradebook_query()`` is ``true``.

gradebook_terms

get_grade_entry_query_record(*grade_entry_record_type*)
Gets the grade entry query record corresponding to the given GradeEntry record Type.

Multiple retrievals produce a nested OR term.

Parameters **grade_entry_record_type** (osid.type.Type) – a grade entry record type

Returns the grade entry query record

Return type osid.grading.records.GradeEntryQueryRecord

Raise NullArgument – grade_entry_record_type is null

Raise OperationFailed – unable to complete request

Raise Unsupported – has_record_type(grade_entry_record_type) is false

compliance: mandatory – This method must be implemented.

Gradebook Column Query

class `dlkit.grading.queries.`**`GradebookColumnQuery`**

Bases: *`dlkit.osid.queries.OsidObjectQuery`*

This is the query for searching gradings.

Each method match request produces an `AND` term while multiple invocations of a method produces a nested `OR`.

`match_grade_system_id`(*grade_system_id*, *match*)

Sets the grade system `Id` for this query.

Parameters

- **`grade_system_id`** (`osid.id.Id`) – a grade system `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `grade_system_id` is `null`

compliance: mandatory – This method must be implemented.

`grade_system_id_terms`

`supports_grade_system_query`()

Tests if a `GradeSystemQuery` is available for querying grade systems.

Returns `true` if a grade system query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`grade_system_query`

Gets the query for a grade system.

Multiple retrievals produce a nested `OR` term.

Returns the grade system query

Return type `osid.grading.GradeSystemQuery`

Raise `Unimplemented` – `supports_grade_system_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_grade_system_query()‘‘ is ‘‘true‘‘.

`match_any_grade_system`(*match*)

Matches gradebook columns with any grade system assigned.

Parameters **`match`** (`boolean`) – `true` to match columns with any grade system, `false` to match columns with no grade system

compliance: mandatory – This method must be implemented.

`grade_system_terms`

`match_grade_entry_id`(*grade_entry_id*, *match*)

Sets the grade entry `Id` for this query.

Parameters

- **`grade_entry_id`** (`osid.id.Id`) – a grade entry `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `grade_entry_id` is `null`

compliance: mandatory – This method must be implemented.

grade_entry_id_terms

supports_grade_entry_query()

Tests if a `GradeEntryQuery` is available for querying grade entries.

Returns `true` if a grade entry query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

grade_entry_query

Gets the query for a grade entry.

Multiple retrievals produce a nested `OR` term.

Returns the grade entry query

Return type `osid.grading.GradeEntryQuery`

Raise `Unimplemented` – `supports_grade_entry_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_grade_entry_query()‘‘ is ‘‘true‘‘.

match_any_grade_entry(*match*)

Matches gradebook columns with any grade entry assigned.

Parameters match (`boolean`) – `true` to match columns with any grade entry, `false` to match columns with no grade entries

compliance: mandatory – This method must be implemented.

grade_entry_terms

supports_gradebook_column_summary_query()

Tests if a `GradebookColumnSummaryQuery` is available for querying grade systems.

Returns `true` if a gradebook column summary query interface is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

gradebook_column_summary_query

Gets the query interface for a gradebook column summary.

Multiple retrievals produce a nested `OR` term.

Returns the gradebook column summary query

Return type `osid.grading.GradebookColumnSummaryQuery`

Raise `Unimplemented` – `supports_gradebook_column_summary_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_gradebook_column_summary_query()‘‘ is ‘‘true‘‘.

gradebook_column_summary_terms

match_gradebook_id(*gradebook_id*, *match*)

Sets the gradebook `Id` for this query.

Parameters

- **gradebook_id** (`osid.id.Id`) – a gradebook `Id`

- **match** (boolean) – true for a positive match, false for a negative match

Raise NullArgument – gradebook_id is null

compliance: mandatory – This method must be implemented.

gradebook_id_terms

supports_gradebook_query()

Tests if a GradebookQuery is available for querying grade systems.

Returns true if a gradebook query interface is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

gradebook_query

Gets the query interface for a gradebook.

Multiple retrievals produce a nested OR term.

Returns the gradebook query

Return type osid.grading.GradebookQuery

Raise Unimplemented – supports_gradebook_query() is false

compliance: optional – This method must be implemented if ‘‘supports_gradebook_query()‘‘ is ‘‘true‘‘.

gradebook_terms

get_gradebook_column_query_record(*gradebook_column_record_type*)

Gets the gradebook column query record corresponding to the given GradebookColumn record Type.

Multiple retrievals produce a nested OR term.

Parameters **gradebook_column_record_type** (osid.type.Type) – a gradebook column record type

Returns the gradebook column query record

Return type osid.grading.records.GradebookColumnQueryRecord

Raise NullArgument – gradebook_column_record_type is null

Raise OperationFailed – unable to complete request

Raise Unsupported – has_record_type(gradebook_column_record_type) is false

compliance: mandatory – This method must be implemented.

Gradebook Column Summary Query

class dlkit.grading.queries.**GradebookColumnSummaryQuery**

Bases: *dlkit.osid.queries.OsidRuleQuery*

This is the query for searching gradebook column summaries.

Each method match request produces an AND term while multiple invocations of a method produces a nested OR.

match_gradebook_column_id(*gradebook_column_id*, *match*)

Sets the gradebook column Id for this query.

Parameters

- **gradebook_column_id** (osid.id.Id) – a gradeboo column Id
- **match** (boolean) – true for a positive match, false for a negative match

Raise NullArgument – gradebook_column_id is null

compliance: mandatory – This method must be implemented.

gradebook_column_id_terms

supports_gradebook_column_query()

Tests if a GradebookColumnQuery is available for querying gradebook column.

Returns true if a gradebook column query is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

gradebook_column_query

Gets the query for a gradebook column.

Multiple retrievals produce a nested OR term.

Returns the gradebook column query

Return type osid.grading.GradebookColumnQuery

Raise Unimplemented – supports_gradebook_column_query() is false

compliance: optional – This method must be implemented if ‘‘supports_gradebook_column_query()‘‘ is ‘‘true‘‘.

match_any_gradebook_column(*match*)

Matches gradebook column derivations with any gradebookc olumn.

Parameters match (boolean) – true to match gradebook column derivations with any gradebook column, false to match gradebook column derivations with no gradebook columns

compliance: mandatory – This method must be implemented.

gradebook_column_terms

match_mean(*low*, *high*, *match*)

Matches a mean between the given values inclusive.

Parameters

- **low** (decimal) – low end of range
- **high** (decimal) – high end of range
- **match** (boolean) – true for a positive match, false for a negative match

Raise InvalidArgument – low is greater than high

compliance: mandatory – This method must be implemented.

mean_terms

match_minimum_mean(*value*, *match*)

Matches a mean greater than or equal to the given value.

Parameters

- **value** (decimal) – minimum value
- **match** (boolean) – true for a positive match, false for a negative match

compliance: mandatory – This method must be implemented.

minimum_mean_terms

match_median (*low*, *high*, *match*)
Matches a median between the given values inclusive.

Parameters

- **low** (`decimal`) – low end of range
- **high** (`decimal`) – high end of range
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `low` is greater than `high`

compliance: mandatory – This method must be implemented.

median_terms

match_minimum_median (*value*, *match*)
Matches a median greater than or equal to the given value.

Parameters

- **value** (`decimal`) – minimum value
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

compliance: mandatory – This method must be implemented.

minimum_median_terms

match_mode (*low*, *high*, *match*)
Matches a mode between the given values inclusive.

Parameters

- **low** (`decimal`) – low end of range
- **high** (`decimal`) – high end of range
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `low` is greater than `high`

compliance: mandatory – This method must be implemented.

mode_terms

match_minimum_mode (*value*, *match*)
Matches a mode greater than or equal to the given value.

Parameters

- **value** (`decimal`) – minimum value
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

compliance: mandatory – This method must be implemented.

minimum_mode_terms

match_rms (*low*, *high*, *match*)
Matches a root mean square between the given values inclusive.

Parameters

- **low** (`decimal`) – low end of range

- **high** (decimal) – high end of range
- **match** (boolean) – true for a positive match, false for a negative match

Raise InvalidArgument – low is greater than high

compliance: mandatory – This method must be implemented.

rms_terms

match_minimum_rms (*value*, *match*)
Matches a root mean square greater than or equal to the given value.

Parameters

- **value** (decimal) – minimum value
- **match** (boolean) – true for a positive match, false for a negative match

compliance: mandatory – This method must be implemented.

minimum_rms_terms

match_standard_deviation (*low*, *high*, *match*)
Matches a standard deviation mean square between the given values inclusive.

Parameters

- **low** (decimal) – low end of range
- **high** (decimal) – high end of range
- **match** (boolean) – true for a positive match, false for a negative match

Raise InvalidArgument – low is greater than high

compliance: mandatory – This method must be implemented.

standard_deviation_terms

match_minimum_standard_deviation (*value*, *match*)
Matches a standard deviation greater than or equal to the given value.

Parameters

- **value** (decimal) – minimum value
- **match** (boolean) – true for a positive match, false for a negative match

compliance: mandatory – This method must be implemented.

minimum_standard_deviation_terms

match_sum (*low*, *high*, *match*)
Matches a sum mean square between the given values inclusive.

Parameters

- **low** (decimal) – low end of range
- **high** (decimal) – high end of range
- **match** (boolean) – true for a positive match, false for a negative match

Raise InvalidArgument – low is greater than high

compliance: mandatory – This method must be implemented.

sum_terms

`match_minimum_sum`(*value*, *match*)
Matches a sum greater than or equal to the given value.

Parameters

- **`value`** (`decimal`) – minimum value
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

compliance: mandatory – This method must be implemented.

`minimum_sum_terms`

`match_gradebook_id`(*gradebook_id*, *match*)
Sets the gradebook `Id` for this query.

Parameters

- **`gradebook_id`** (`osid.id.Id`) – a gradebook `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `gradebook_id` is `null`

compliance: mandatory – This method must be implemented.

`gradebook_id_terms`

`supports_gradebook_query`()
Tests if a `GradebookQuery` is available.

Returns `true` if a gradebook query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`gradebook_query`
Gets the query for a gradebook.

Multiple retrievals produce a nested `OR` term.

Returns the gradebook query

Return type `osid.grading.GradebookQuery`

Raise `Unimplemented` – `supports_gradebook_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_gradebook_column_query()‘‘ is ‘‘true‘‘.

`gradebook_terms`

`get_gradebook_column_summary_query_record`(*gradebook_column_summary_record_type*)
Gets the gradebook column summary query record corresponding to the given `GradebookColumnSummary` record `Type`.

Multiple retrievals produce a nested `OR` term.

Parameters **`gradebook_column_summary_record_type`** (`osid.type.Type`) – a gradebook column summary record type

Returns the gradebook column summary query record

Return type `osid.grading.records.GradebookColumnSummaryQueryRecord`

Raise `NullArgument` – `gradebook_column_summary_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(gradebook_column_summary_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Gradebook Query

class `dlkit.grading.queries.`**`GradebookQuery`**

Bases: *`dlkit.osid.queries.OsidCatalogQuery`*

This is the query for searching gradebooks.

Each method specifies an `AND` term while multiple invocations of the same method produce a nested `OR`.

`match_grade_system_id`(*grade_system_id*, *match*)

Sets the grade system `Id` for this query.

Parameters

- **`grade_system_id`** (`osid.id.Id`) – a grade system `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `grade_system_id` is `null`

compliance: mandatory – This method must be implemented.

`grade_system_id_terms`

`supports_grade_system_query`()

Tests if a `GradeSystemQuery` is available.

Returns `true` if a grade system query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`grade_system_query`

Gets the query for a grade system.

Multiple retrievals produce a nested `OR` term.

Returns the grade system query

Return type `osid.grading.GradeSystemQuery`

Raise `Unimplemented` – `supports_grade_system_query()` is `false`

compliance: optional – This method must be implemented if ``supports_grade_system_query()`` is ``true``.

`match_any_grade_system`(*match*)

Matches gradebooks that have any grade system.

Parameters **`match`** (`boolean`) – `true` to match gradebooks with any grade system, `false` to match gradebooks with no grade system

compliance: mandatory – This method must be implemented.

`grade_system_terms`

`match_grade_entry_id`(*grade_entry_id*, *match*)

Sets the grade entry `Id` for this query.

Parameters

- **`grade_entry_id`** (`osid.id.Id`) – a grade entry `Id`

- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `grade_entry_id` is `null`

compliance: mandatory – This method must be implemented.

`grade_entry_id_terms`

`supports_grade_entry_query``()`
Tests if a `GradeEntryQuery` is available.

Returns `true` if a grade entry query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`grade_entry_query`
Gets the query for a grade entry.

Multiple retrievals produce a nested `OR` term.

Returns the grade entry query

Return type `osid.grading.GradeEntryQuery`

Raise `Unimplemented` – `supports_grade_entry_query()` is `false`

compliance: optional – This method must be implemented if ``supports_grade_entry_query()`` is ``true``.

`match_any_grade_entry``(`*match*`)`
Matches gradebooks that have any grade entry.

Parameters **match** (`boolean`) – `true` to match gradebooks with any grade entry, `false` to match gradebooks with no grade entry

compliance: mandatory – This method must be implemented.

`grade_entry_terms`

`match_gradebook_column_id``(`*gradebook_column_id*, *match*`)`
Sets the gradebook column `Id` for this query.

Parameters

- **gradebook_column_id** (`osid.id.Id`) – a gradebook column `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `gradebook_column_id` is `null`

compliance: mandatory – This method must be implemented.

`gradebook_column_id_terms`

`supports_gradebook_column_query``()`
Tests if a `GradebookColumnQuery` is available.

Returns `true` if a gradebook column query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`gradebook_column_query`
Gets the query for a gradebook column.

Multiple retrievals produce a nested `OR` term.

Returns the gradebook column query

Return type `osid.grading.GradebookColumnQuery`

Raise `Unimplemented` – `supports_gradebook_column_query()` is `false`

compliance: optional – This method must be implemented if ``supports_gradebook_column_query()`` is ``true``.

`match_any_gradebook_column`(*match*)

Matches gradebooks that have any column.

Parameters **`match`** (`boolean`) – `true` to match gradebooks with any column, `false` to match gradebooks with no column

compliance: mandatory – This method must be implemented.

`gradebook_column_terms`

`match_ancestor_gradebook_id`(*gradebook_id*, *match*)

Sets the gradebook `Id` for this query to match gradebooks that have the specified gradebook as an ancestor.

Parameters

- **`gradebook_id`** (`osid.id.Id`) – a gradebook `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `gradebook_id` is `null`

compliance: mandatory – This method must be implemented.

`ancestor_gradebook_id_terms`

`supports_ancestor_gradebook_query`()

Tests if a `GradebookQuery` is available.

Returns `true` if a gradebook query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ancestor_gradebook_query`

Gets the query for a gradebook.

Multiple retrievals produce a nested `OR` term.

Returns the gradebook query

Return type `osid.grading.GradebookQuery`

Raise `Unimplemented` – `supports_ancestor_gradebook_query()` is `false`

compliance: optional – This method must be implemented if ``supports_ancestor_gradebook_query()`` is ``true``.

`match_any_ancestor_gradebook`(*match*)

Matches gradebook with any ancestor.

Parameters **`match`** (`boolean`) – `true` to match gradebooks with any ancestor, `false` to match root gradebooks

compliance: mandatory – This method must be implemented.

`ancestor_gradebook_terms`

match_descendant_gradebook_id(*gradebook_id*, *match*)

Sets the gradebook `Id` for this query to match gradebooks that have the specified gradebook as a descendant.

Parameters

- **gradebook_id** (`osid.id.Id`) – a gradebook `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `gradebook_id` is `null`

compliance: mandatory – This method must be implemented.

descendant_gradebook_id_terms

supports_descendant_gradebook_query()

Tests if a `GradebookQuery` is available.

Returns `true` if a gradebook query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

descendant_gradebook_query

Gets the query for a gradebook.

Multiple retrievals produce a nested `OR` term.

Returns the gradebook query

Return type `osid.grading.GradebookQuery`

Raise `Unimplemented` – `supports_descendant_gradebook_query()` is `false`

compliance: optional – This method must be implemented if ``supports_descendant_gradebook_query()`` is ``true``.

match_any_descendant_gradebook(*match*)

Matches gradebook with any descendant.

Parameters **match** (`boolean`) – `true` to match gradebooks with any descendant, `false` to match leaf gradebooks

compliance: mandatory – This method must be implemented.

descendant_gradebook_terms

get_gradebook_query_record(*gradebook_record_type*)

Gets the gradebook query record corresponding to the given `Gradebook` record `Type`.

Multiple record retrievals produce a nested `OR` term.

Parameters **gradebook_record_type** (`osid.type.Type`) – a gradebook record type

Returns the gradebook query record

Return type `osid.grading.records.GradebookQueryRecord`

Raise `NullArgument` – `gradebook_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(gradebook_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Records

Grade Record

class `dlkit.grading.records.`**`GradeRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `Grade`.

The methods specified by the record type are available through the underlying object.

Grade Query Record

class `dlkit.grading.records.`**`GradeQueryRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `GradeQuery`.

The methods specified by the record type are available through the underlying object.

Grade Form Record

class `dlkit.grading.records.`**`GradeFormRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `GradeForm`.

The methods specified by the record type are available through the underlying object.

Grade System Record

class `dlkit.grading.records.`**`GradeSystemRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `GradeSystem`.

The methods specified by the record type are available through the underlying object.

Grade System Query Record

class `dlkit.grading.records.`**`GradeSystemQueryRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `GradeSystemQuery`.

The methods specified by the record type are available through the underlying object.

Grade System Form Record

class `dlkit.grading.records.`**`GradeSystemFormRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `GradeSystemForm`.

The methods specified by the record type are available through the underlying object.

Grade System Search Record

class `dlkit.grading.records.`**`GradeSystemSearchRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `GradeSystemSearch`.

The methods specified by the record type are available through the underlying object.

Grade Entry Record

class `dlkit.grading.records.`**`GradeEntryRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `GradeEntry`.

The methods specified by the record type are available through the underlying object.

Grade Entry Query Record

class `dlkit.grading.records.`**`GradeEntryQueryRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `GradeEntryQuery`.

The methods specified by the record type are available through the underlying object.

Grade Entry Form Record

class `dlkit.grading.records.`**`GradeEntryFormRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `GradeEntryForm`.

The methods specified by the record type are available through the underlying object.

Grade Entry Search Record

class `dlkit.grading.records.`**`GradeEntrySearchRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `GradeEntrySearch`.

The methods specified by the record type are available through the underlying object.

Gradebook Column Record

class `dlkit.grading.records.`**`GradebookColumnRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `GradebookColumn`.

The methods specified by the record type are available through the underlying object.

Gradebook Column Query Record

class `dlkit.grading.records.`**`GradebookColumnQueryRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `GradebookColumnQuery`.

The methods specified by the record type are available through the underlying object.

Gradebook Column Form Record

class `dlkit.grading.records.`**`GradebookColumnFormRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `GradebookColumnForm`.

The methods specified by the record type are available through the underlying object.

Gradebook Column Search Record

class `dlkit.grading.records.`**`GradebookColumnSearchRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `GradebookColumnSearch`.

The methods specified by the record type are available through the underlying object.

Gradebook Column Summary Record

class `dlkit.grading.records.`**`GradebookColumnSummaryRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `GradebookColumnSummary`.

The methods specified by the record type are available through the underlying object.

Gradebook Column Summary Query Record

class `dlkit.grading.records.`**`GradebookColumnSummaryQueryRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `GradebookColumnSummaryQuery`.

The methods specified by the record type are available through the underlying object.

Gradebook Record

class `dlkit.grading.records.`**`GradebookRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `Gradebook`.

The methods specified by the record type are available through the underlying object.

Gradebook Query Record

class `dlkit.grading.records.`**`GradebookQueryRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `GradebookQuery`.

The methods specified by the record type are available through the underlying object.

Gradebook Form Record

class `dlkit.grading.records.`**`GradebookFormRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `GradebookForm`.

The methods specified by the record type are available through the underlying object.

Gradebook Search Record

class `dlkit.grading.records.`**`GradebookSearchRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `GradebookSearch`.

The methods specified by the record type are available through the underlying object.

Hierarchy

Summary

Hierarchy Open Service Interface Definitions hierarchy version 3.0.0

The Hierarchy OSID is an auxiliary service providing a means for accessing and managing hierarchical relationships among OSID `Ids`.

An OSID `Id` may have onr or more parents or children and the hierarchy itself represents a directed acyclic graph. The hierarchy service defines a set of interfaces used among other OSIDs that utilize hierarchies and can also be used to abstract hierarchical data into a standalone service.

Hierarchical queries may be performed using the `HierarchyTraversalSession`. A set of methods exist to query parents, children, ancestors, and decendants. A Node structure may be retrieved to access a portion of a hierarchy in bulk. The `Node` provides methods to get parents and children of the node directly.

Hierarchies are federateable by combining nodes. There is no hierarchy service for the hierarchy catalog. Hierarchy Open Service Interface Definitions hierarchy version 3.0.0

The Hierarchy OSID is an auxiliary service providing a means for accessing and managing hierarchical relationships among OSID `Ids`.

An OSID `Id` may have onr or more parents or children and the hierarchy itself represents a directed acyclic graph. The hierarchy service defines a set of interfaces used among other OSIDs that utilize hierarchies and can also be used to abstract hierarchical data into a standalone service.

Hierarchical queries may be performed using the `HierarchyTraversalSession`. A set of methods exist to query parents, children, ancestors, and decendants. A Node structure may be retrieved to access a portion of a hierarchy in bulk. The `Node` provides methods to get parents and children of the node directly.

Hierarchies are federateable by combining nodes. There is no hierarchy service for the hierarchy catalog.

Service Managers

Hierarchy Profile

class `dlkit.services.hierarchy.`**`HierarchyProfile`**

Bases: `dlkit.osid.managers.OsidProfile`

The hierarchy profile describes the interoperability among hierarchy services.

`supports_hierarchy_traversal``()`

Tests if hierarchy traversal is supported.

Returns `true` if hierarchy traversal is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_hierarchy_design``()`

Tests if hierarchy design is supported.

Returns `true` if hierarchy design is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_hierarchy_lookup``()`

Tests if a hierarchy lookup is supported.

Returns `true` if hierarchy lookup is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_hierarchy_admin``()`

Tests if a hierarchy administration is supported.

Returns `true` if hierarchy administration is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`hierarchy_record_types`

Gets the supported `Hierarchy` types.

Returns a list containing the supported `Hierarchy` record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

`hierarchy_search_record_types`

Gets the supported `Hierarchy` search record types.

Returns a list containing the supported `Hierarchy` search record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

Hierarchy Manager

class `dlkit.services.hierarchy.`**`HierarchyManager`**(*proxy=None*)

Bases: `dlkit.osid.managers.OsidManager`, `dlkit.osid.sessions.OsidSession`, *`dlkit.services.hierarchy.HierarchyProfile`*

The hierarchy manager provides access sessions to traverse and manage hierrachies of `Ids`.

The sessions included in this manager are:

- `HierarchyTraversalSession`: a basic session traversing a hierarchy
- `HierarchyDesignSession`: a session to design a hierarchy
- `HierarchySequencingSession`: a session to sequence nodes in a hierarchy
- `HierarchyStructureNotificationSession`: a session for notififcations within a hierarchy structure
- `HierarchyLookupSession`: a session looking up hiererachies
- `HierarchyQuerySession`: a session querying hiererachies
- `HierarchySearchSession`: a session for searching for hierarchies
- `HierarchyAdminSession`: a session for creating and deleting hierarchies
- `HierarchyNotificationSession`: a session for subscribing to changes in hierarchies

Hierarchy Traversal Methods

`HierarchyManager.`**`hierarchy_id`**

Gets the hierarchy `Id` associated with this session.

Returns the hierarchy `Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`hierarchy`**

Gets the hierarchy associated with this session.

Returns the hierarchy associated with this session

Return type `osid.hierarchy.Hierarchy`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`can_access_hierarchy`**()

Tests if this user can perform hierarchy queries.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations.

Returns `false` if lookup methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`roots`**

Gets the root nodes of this hierarchy.

Returns the root nodes

Return type `osid.id.IdList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`has_parents`**`(`*id_*`)`

Tests if this `Id` contains any parents.

Parameters **`id`** (`osid.id.Id`) – the `Id` to query

Returns `true` if this `Id` contains parents, `false` otherwise

Return type `boolean`

Raise `NotFound` – `id` is not found

Raise `NullArgument` – `id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`is_parent`**`(`*id_*, *parent_id*`)`

Tests if an `Id` is a direct parent of another.

Parameters

- **`id`** (`osid.id.Id`) – the `Id` to query
- **`parent_id`** (`osid.id.Id`) – the `Id` of a parent

Returns `true` if this `parent_id` is a parent of `id, false` otherwise

Return type `boolean`

Raise `NotFound` – `id` is not found

Raise `NullArgument` – `id` or `parent_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `parent_id` not found return `false`.

`HierarchyManager.`**`get_parents`**`(`*id_*`)`

Gets the parents of the given `id`.

Parameters **`id`** (`osid.id.Id`) – the `Id` to query

Returns the parents of the `id`

Return type `osid.id.IdList`

Raise `NotFound` – `id` is not found

Raise `NullArgument` – `id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`is_ancestor`**`(`*id_*, *ancestor_id*`)`

Tests if an `Id` is an ancestor of another.

Parameters

- **`id`** (`osid.id.Id`) – the `Id` to query
- **`ancestor_id`** (`osid.id.Id`) – the `Id` of an ancestor

Returns `true` if this `ancestor_id` is a parent of `id, false` otherwise

Return type `boolean`

Raise `NotFound` – `id` is not found

Raise `NullArgument` – `id` or `ancestor_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `ancestor_id` not found return `false`.

`HierarchyManager.`**`has_children`**`(`*id_*`)`

Tests if this `Id` has any children.

Parameters **`id`** (`osid.id.Id`) – the `Id` to query

Returns `true` if this `Id` has children, `false` otherwise

Return type `boolean`

Raise `NotFound` – `id` is not found

Raise `NullArgument` – `id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`is_child`**`(`*id_*, *child_id*`)`

Tests if a node is a direct child of another.

Parameters

- **`id`** (`osid.id.Id`) – the `Id` to query
- **`child_id`** (`osid.id.Id`) – the `Id` of a child

Returns `true` if this `child_id` is a child of the `Id, false` otherwise

Return type `boolean`

Raise `NotFound` – `id` is not found

Raise `NullArgument` – `id` or `child_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `child_id` not found return `false`.

`HierarchyManager.`**`get_children`**`(`*id_*`)`

Gets the children of the given `Id`.

Parameters **`id`** (`osid.id.Id`) – the `Id` to query

Returns the children of the `id`

Return type `osid.id.IdList`

Raise `NotFound` – `id` is not found

Raise `NullArgument` – `id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`is_descendant`**`(`*id_*, *descendant_id*`)`

Tests if a node is a descendant of another.

Parameters

- **`id`** (`osid.id.Id`) – the `Id` to query
- **`descendant_id`** (`osid.id.Id`) – the `Id` of a descendant

Returns `true` if this `descendant_id` is a child of the `Id, false` otherwise

Return type `boolean`

Raise `NotFound` – `id` is not found

Raise `NullArgument` – `id` or `descendant` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If not found return `false`.

`HierarchyManager.`**`get_nodes`**`(`*id_*, *ancestor_levels*, *descendant_levels*, *include_siblings*`)`

Gets a portion of the hierarchy for the given `Id`.

Parameters

- **`id`** (`osid.id.Id`) – the `Id` to query
- **`ancestor_levels`** (`cardinal`) – the maximum number of ancestor levels to include. A value of 0 returns no parents in the node.
- **`descendant_levels`** (`cardinal`) – the maximum number of descendant levels to include. A value of 0 returns no children in the node.
- **`include_siblings`** (`boolean`) – `true` to include the siblings of the given node, `false` to omit the siblings

Returns a node

Return type `osid.hierarchy.Node`

Raise `NotFound` – `id` is not found

Raise `NullArgument` – `id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Hierarchy Design Methods

`HierarchyManager.`**`hierarchy_id`**

Gets the hierarchy `Id` associated with this session.

Returns the hierarchy `Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`hierarchy`**

Gets the hierarchy associated with this session.

Returns the hierarchy associated with this session

Return type `osid.hierarchy.Hierarchy`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`can_modify_hierarchy`**`()`

Tests if this user can change the hierarchy.

A return of true does not guarantee successful authorization. A return of false indicates that it is known performing any update will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer these operations to an unauthorized user.

Returns `false` if changing this hierarchy is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`add_root`**`(`*id_*`)`

Adds a root node.

Parameters **`id`** (`osid.id.Id`) – the `Id` of the node

Raise `AlreadyExists` – `id` is already in hierarchy

Raise `NotFound` – `id` not found

Raise `NullArgument` – `id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`add_child`**`(`*id_*, *child_id*`)`

Adds a child to a `Id`.

Parameters

- **`id`** (`osid.id.Id`) – the `Id` of the node
- **`child_id`** (`osid.id.Id`) – the `Id` of the new child

Raise `AlreadyExists` – `child_id` is already a child of `id`

Raise `NotFound` – `id` or `child_id` not found

Raise `NullArgument` – `id` or `child_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`remove_root`**`(`*id_*`)`

Removes a root node.

Parameters **`id`** (`osid.id.Id`) – the `Id` of the node

Raise `NotFound` – `id` was not found or not in hierarchy

Raise `NullArgument` – `id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`remove_child`**`(`*id_*, *child_id*`)`

Removes a childfrom an `Id`.

Parameters

- **`id`** (`osid.id.Id`) – the `Id` of the node
- **`child_id`** (`osid.id.Id`) – the `Id` of the child to remove

Raise `NotFound` – `id` or `child_id` was not found or `child_id` is not a child of `id`

Raise `NullArgument` – `id` or `child_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`remove_children`**`(`*id_*`)`

Removes all childrenfrom an `Id`.

Parameters **`id`** (`osid.id.Id`) – the `Id` of the node

Raise `NotFound` – an node identified by the given `Id` was not found

Raise `NullArgument` – `id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Hierarchy Lookup Methods

`HierarchyManager.`**`can_lookup_hierarchies`**`()`

Tests if this user can perform `Hierarchy` lookups.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if lookup methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`use_comparative_hierarchy_view`**`()`

The returns from the lookup methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`HierarchyManager.`**`use_plenary_hierarchy_view`**`()`

A complete view of the `Hierarchy` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`HierarchyManager.`**`get_hierarchy`**`(`*hierarchy_id*`)`

Gets the `Hierarchy` specified by its `Id`.

In plenary mode, the exact `Id` is found or a `NotFound` results. Otherwise, the returned `Hierarchy` may have a different `Id` than requested, such as the case where a duplicate `Id` was assigned to a `Hierarchy` and retained for compati

Parameters **`hierarchy_id`** (`osid.id.Id`) – the `Id` of the `Hierarchy` to retrieve

Returns the returned `Hierarchy`

Return type `osid.hierarchy.Hierarchy`

Raise `NotFound` – no `Hierarchy` found with the given `Id`

Raise `NullArgument` – `hierarchy_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`get_hierarchies_by_ids`**`(`*hierarchy_ids*`)`

Gets a `Hierarchy` corresponding to the given `IdList`.

In plenary mode, the returned list contains all of the hierarchies specified in the `Id` list, in the order of the list, including duplicates, or an error results if an `Id` in the supplied list is not found or inaccessible. Otherwise, inaccessible `Hierarchy` objects may be omitted from the list and may present the elements in any order including returning a unique set.

Parameters **`hierarchy_ids`** (`osid.id.IdList`) – the list of `Ids` to retrieve

Returns the returned `Hierarchy` list

Return type `osid.hierarchy.HierarchyList`

Raise `NotFound` – an `Id was` not found

Raise `NullArgument` – `hierarchy_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`get_hierarchies_by_genus_type`**(*hierarchy_genus_type*)

Gets a `HierarchyList` corresponding to the given genus `Type` which does not include hierarchies of types derived from the specified `Type`.

In plenary mode, the returned list contains all known hierarchies or an error results. Otherwise, the returned list may contain only those hierarchies that are accessible through this session.

Parameters **`hierarchy_genus_type`** (`osid.type.Type`) – a hierarchy genus type

Returns the returned `Hierarchy` list

Return type `osid.hierarchy.HierarchyList`

Raise `NullArgument` – `hierarchy_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`get_hierarchies_by_parent_genus_type`**(*hierarchy_genus_type*)

Gets a `HierarchyList` corresponding to the given hierarchy genus `Type` and include any additional hierarchies with types derived from the specified `Type`.

In plenary mode, the returned list contains all known hierarchies or an error results. Otherwise, the returned list may contain only those hierarchies that are accessible through this session.

Parameters **`hierarchy_genus_type`** (`osid.type.Type`) – a hierarchy genus type

Returns the returned `Hierarchy` list

Return type `osid.hierarchy.HierarchyList`

Raise `NullArgument` – `hierarchy_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`get_hierarchies_by_record_type`**(*hierarchy_record_type*)

Gets a `HierarchyList` corresponding to the given hierarchy record `Type`.

The set of hierarchies implementing the given record type are returned.In plenary mode, the returned list contains all known hierarchies or an error results. Otherwise, the returned list may contain only those hierarchies that are accessible through this session.

Parameters **`hierarchy_record_type`** (`osid.type.Type`) – a hierarchy record type

Returns the returned `Hierarchy` list

Return type `osid.hierarchy.HierarchyList`

Raise `NullArgument` – `hierarchy_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`get_hierarchies_by_provider`**`(`*resource_id*`)`

Gets a `HierarchyList` for the given provider ````.

The set of hierarchies implementing the given record type are returned.In plenary mode, the returned list contains all known hierarchies or an error results. Otherwise, the returned list may contain only those hierarchies that are accessible through this session.

Parameters **`resource_id`** (`osid.id.Id`) – a resource `Id`

Returns the returned `Hierarchy` list

Return type `osid.hierarchy.HierarchyList`

Raise `NullArgument` – `resource_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`hierarchies`**

Gets all hierarchies.

In plenary mode, the returned list contains all known hierarchies or an error results. Otherwise, the returned list may contain only those hierarchies that are accessible through this session.

Returns a list of `Hierarchies`

Return type `osid.hierarchy.HierarchyList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Hierarchy Admin Methods

`HierarchyManager.`**`can_create_hierarchies`**`()`

Tests if this user can create `Hierarchy` objects.

A return of true does not guarantee successful authorization. A return of false indicates that it is known creating a `Hierarchy` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer create operations to unauthorized users.

Returns `false` if `Hierarchy` creation is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`can_create_hierarchy_with_record_types`**`(`*hierarchy_record_types*`)`

Tests if this user can create a single `Hierarchy` using the desired record types.

While `HierarchyManager.getHierarchyRecordTypes()` can be used to examine which records are supported, this method tests which record(s) are required for creating a specific `Hierarchy`. Providing an empty array tests if a `Hierarchy` can be created with no records.

Parameters **`hierarchy_record_types`** (`osid.type.Type[]`) – array of hierarchy record types

Returns `true` if `Hierarchy` creation using the specified `Types` is supported, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `hierarchy_record_types` is `null`

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`get_hierarchy_form_for_create`**`(`*hierarchy_record_types*`)`

Gets the hierarchy form for creating new hierarchies.

A new form should be requested for each create transaction. This method is used for creating new hierarchies, where only the `Hierarchy Type` is known.

Parameters **`hierarchy_record_types`** (`osid.type.Type[]`) – array of hierarchy record types

Returns the hierarchy form

Return type `osid.hierarchy.HierarchyForm`

Raise `NullArgument` – `hierarchy_record_types` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – unable to get form for requested record types

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`create_hierarchy`**`(`*hierarchy_form*`)`

Creates a new `Hierarchy`.

Parameters **`hierarchy_form`** (`osid.hierarchy.HierarchyForm`) – the form for this `Hierarchy`

Returns the new `Hierarchy`

Return type `osid.hierarchy.Hierarchy`

Raise `IllegalState` – `hierarchy_form` already used in a create transaction

Raise `InvalidArgument` – one or more of the form elements is invalid

Raise `NullArgument` – `hierarchy_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `hierarchy_form` did not originate from `get_hierarchy_form_for_create()`

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`can_update_hierarchies`**`()`

Tests if this user can update `Hierarchy` objects.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating a `Hierarchy` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer update operations to unauthorized users.

Returns `false` if `Hierarchy` modification is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`get_hierarchy_form_for_update`**`(`*hierarchy_id*`)`

Gets the hierarchy form for updating an existing hierarchy.

A new hierarchy form should be requested for each update transaction.

Parameters **`hierarchy_id`** (`osid.id.Id`) – the `Id` of the `Hierarchy`

Returns the hierarchy form

Return type `osid.hierarchy.HierarchyForm`

Raise `NotFound` – `hierarchy_id` is not found

Raise `NullArgument` – `hierarchy_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`update_hierarchy`**`(`*hierarchy_form*`)`

Updates an existing hierarchy.

Parameters **`hierarchy_form`** (`osid.hierarchy.HierarchyForm`) – the form containing the elements to be updated

Raise `IllegalState` – `hierarchy_form` already used in an update transaction

Raise `InvalidArgument` – the form contains an invalid value

Raise `NullArgument` – `hierarchy_id` or `hierarchy_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `hierarchy_form` did not originate from `get_hierarchy_form_for_update()`

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`can_delete_hierarchies`**`()`

Tests if this user can delete `Hierarchy` objects.

A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting a `Hierarchy` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer delete operations to unauthorized users.

Returns `false` if `Hierarchy` deletion is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`HierarchyManager.`**`delete_hierarchy`**`(`*hierarchy_id*`)`

Deletes a `Hierarchy`.

Parameters **`hierarchy_id`** (`osid.id.Id`) – the `Id` of the `Hierarchy` to remove

Raise `NotFound` – `hierarchy_id` not found

Raise `NullArgument` – `hierarchy_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

HierarchyManager.**can_manage_hierarchy_aliases**()

Tests if this user can manage Id aliases for Hierarchy objects.

A return of true does not guarantee successful authorization. A return of false indicates that it is known changing an alias will result in a PermissionDenied. This is intended as a hint to an application that may opt not to offer alias operations to an unauthorized user.

Returns false if Hierarchy aliasing is not authorized, true otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

HierarchyManager.**alias_hierarchy**(*hierarchy_id*, *alias_id*)

Adds an Id to a Hierarchy for the purpose of creating compatibility.

The primary Id of the Hierarchy is determined by the provider. The new Id performs as an alias to the primary Id. If the alias is a pointer to another vault it is reassigned to the given vault Id.

Parameters

- **hierarchy_id** (osid.id.Id) – the Id of an Hierarchy
- **alias_id** (osid.id.Id) – the alias Id

Raise AlreadyExists – alias_id is already assigned

Raise NotFound – hierarchy_id not found

Raise NullArgument – hierarchy_id or alias_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Hierarchy

Hierarchy

class dlkit.services.hierarchy.**Hierarchy**(*provider_manager*, *catalog*, *runtime*, *proxy*, ***kwargs*)

Bases: *dlkit.osid.objects.OsidCatalog*, dlkit.osid.sessions.OsidSession

A Hierarchy represents an authenticatable identity.

Like all OSID objects, a Hierarchy is identified by its Id and any persisted references should use the Id.

get_hierarchy_record(*hierarchy_record_type*)

Gets the hierarchy record corresponding to the given Hierarchy record Type.

This method is used to retrieve an object implementing the requested record. The hierarchy_record_type may be the Type returned in get_record_types() or any of its parents in a Type hierarchy where has_record_type(hierarchy_record_type) is true .

Parameters **hierarchy_record_type** (osid.type.Type) – the type of the record to retrieve

Returns the hierarchy record

Return type osid.hierarchy.records.HierarchyRecord

Raise `NullArgument` – `hierarchy_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(hierarchyrecord_type)` is `false`

compliance: mandatory – This method must be implemented.

Objects

Hierarchy

class `dlkit.hierarchy.objects.`**`Hierarchy`**(*abc_hierarchy_objects.Hierarchy, osid_objects.OsidCatalog*)

`:noindex:`

`get_hierarchy_record`(*hierarchy_record_type*)

Gets the hierarchy record corresponding to the given `Hierarchy` record `Type`.

This method is used to retrieve an object implementing the requested record. The `hierarchy_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(hierarchy_record_type)` is `true` .

Parameters **`hierarchy_record_type`** (`osid.type.Type`) – the type of the record to retrieve

Returns the hierarchy record

Return type `osid.hierarchy.records.HierarchyRecord`

Raise `NullArgument` – `hierarchy_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(hierarchyrecord_type)` is `false`

compliance: mandatory – This method must be implemented.

Hierarchy Form

class `dlkit.hierarchy.objects.`**`HierarchyForm`**

Bases: *`dlkit.osid.objects.OsidCatalogForm`*

This is the form for creating and updating `Hierarchy` objects.

Like all `OsidForm` objects, various data elements may be set here for use in the create and update methods in the `HierarchyAdminSession`. For each data element that may be set, metadata may be examined to provide display hints or data constraints.

`get_hierarchy_form_record`(*hierarchy_record_type*)

Gets the `HierarchyFormRecord` corresponding to the given hierarchy record `Type`.

Parameters **`hierarchy_record_type`** (`osid.type.Type`) – the hierarchy record type

Returns the hierarchy form record

Return type `osid.hierarchy.records.HierarchyFormRecord`

Raise `NullArgument` – `hierarchy_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(hierarchy_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Hierarchy List

class `dlkit.hierarchy.objects.`**`HierarchyList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists`, `HierarchyList` provides a means for accessing `Id` elements sequentially either one at a time or many at a time.

Examples: while (hl.hasNext()) { Hierarchy hierarchy = hl.getNextHierarchy(); }

or

while (hl.hasNext()) { Hierarchy[] hierarchies = hl.getNextHierarchies(hl.available());

}

`next_hierarchy`

Gets the next `Hierarchy` in this list.

Returns the next `Hierarchy` in this list. The `has_next()` method should be used to test that a next `Hierarchy` is available before calling this method.

Return type `osid.hierarchy.Hierarchy`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_hierarchies`(*n*)

Gets the next set of `Hierarchy` objects in this list.

The specified amount must be less than or equal to the return from `available()`.

Parameters **n** (`cardinal`) – the number of `Hierarchy` elements requested which must be less than or equal to `available()`

Returns an array of `Hierarchy` elements.The length of the array is less than or equal to the number specified.

Return type `osid.hierarchy.Hierarchy`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Node

class `dlkit.hierarchy.objects.`**`Node`**

Bases: *`dlkit.osid.objects.OsidNode`*

This interface is a container for a partial hierarchy retrieval.

The number of hierarchy levels traversable through this interface depend on the number of levels requested in the hierarchy traversal session.

parents

Gets the parents of this node.

Returns the parents of this node

Return type `osid.hierarchy.NodeList`

compliance: mandatory – This method must be implemented.

children

Gets the children of this node.

Returns the children of this node

Return type `osid.hierarchy.NodeList`

compliance: mandatory – This method must be implemented.

Node List

class `dlkit.hierarchy.objects.`**`NodeList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists`, `NodeList` provides a means for accessing `Id` elements sequentially either one at a time or many at a time.

Examples: while (nl.hasNext()) { Node node = nl.getNextNode(); }

or

while (nl.hasNext()) { Node[] nodes = nl.getNextNodes(nl.available());

}

next_node

Gets the next `Node` in this list.

Returns the next `Node` in this list. The `has_next()` method should be used to test that a next `Node` is available before calling this method.

Return type `osid.hierarchy.Node`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

get_next_nodes(*n*)

Gets the next set of `Node` objects in this list.

The specified amount must be less than or equal to the return from `available()`.

Parameters **n** (`cardinal`) – the number of `Node` elements requested which must be less than or equal to `available()`

Returns an array of `Node` elements.The length of the array is less than or equal to the number specified.

Return type `osid.hierarchy.Node`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Queries

Hierarchy Query

class `dlkit.hierarchy.queries.`**`HierarchyQuery`**

Bases: *`dlkit.osid.queries.OsidCatalogQuery`*

This is the query for searching hierarchies.

Results are returned if all the specified elements match. Each method match request produces an `AND` term while multiple invocations of a method produces a nested `OR`, except for accessing the `HierarchyQuery` record.

`match_node_id`(*id_*, *match*)

Matches an `Id` of a node in this hierarchy.

Multiple nodes can be added to this query which behave as a boolean `AND`.

Parameters

- **`id`** (`osid.id.Id`) – `Id` to match
- **`match`** (`boolean`) – `true` if a positive match, `false` for negative match

Raise `NullArgument` – `id` is `null`

compliance: mandatory – This method must be implemented.

`match_any_node_id`(*match*)

Matches hierarchies with any node.

Parameters **`match`** (`boolean`) – `true` to match hierarchies with any nodes, `false` to match hierarchies with no nodes

compliance: mandatory – This method must be implemented.

`node_id_terms`

`get_hierarchy_query_record`(*hierarchy_record_type*)

Gets the hierarchy record query corresponding to the given `Hierarchy` record `Type`.

Multiple record retrievals of the same type may return the same underlying object and do not result in adding terms to the query. Multiple record retrievals of different types add `AND` terms to the other elements set in this form.

Parameters **`hierarchy_record_type`** (`osid.type.Type`) – a hierarchy record type

Returns the hierarchy query record

Return type `osid.hierarchy.records.HierarchyQueryRecord`

Raise `NullArgument` – `hierarchy_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(hierarchy_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Records

Hierarchy Record

class `dlkit.hierarchy.records.`**`HierarchyRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `Hierarchy`.

The methods specified by the record type are available through the underlying object.

Hierarchy Query Record

class `dlkit.hierarchy.records.`**`HierarchyQueryRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `HierarchyQuery`.

The methods specified by the record type are available through the underlying object.

Hierarchy Form Record

class `dlkit.hierarchy.records.`**`HierarchyFormRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `HierarchyForm`.

The methods specified by the record type are available through the underlying object.

Hierarchy Search Record

class `dlkit.hierarchy.records.`**`HierarchySearchRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `HierarchySearch`.

The methods specified by the record type are available through the underlying object.

Learning

Summary

Learning Open Service Interface Definitions learning version 3.0.0

The Learning OSID manages learning objectives. A learning `Objective` describes measurable learning goals.

Objectives

`Objectives` describe measurable learning goals. A learning objective may be measured by a related `Assesment`. `Objectives` may be mapped to levels, A level is represented by a `Grade` which is used to indicate a grade level or level of difficulty.

`Objectives` are hierarchical. An `Objective` with children represents an objective that is inclusive of all its children. For example, an `Objective` that represents learning in arithmetic may be composed of objectives that represent learning in both addition and subtraction.

`Objectives` may also have requisites. A requisite objective is one that should be achieved before an objective is attempted.

Activities

An `Activity` describes actions that one can do to meet a learning objective. An `Activity` includes a list of `Assets` to read or watch, or a list of `Courses` to take, or a list of learning `Assessments` to practice. An `Activity` may also represent other learning activities such as taking a course or practicing an instrument. An `Activity` is specific to an `Objective` where the reusability is achieved based on what the `Activity` relates.

Proficiencies

A `Proficiency` is an `OsidRelationship` measuring the competence of a `Resource` with respect to an Objective.

Objective Bank Cataloging

`Objectives, Activities,` and `Proficiencies` can be organized into hierarchical `ObjectiveBanks` for the purposes of categorization and federation.

Concept Mapping

A concept can be modeled as a learning `Objective` without any related `Assessment` or `Activities`. In this scenario, an `Objective` looks much like the simpler `Subject` in the Ontology OSID. The Ontology OSID is constrained to qualifying concepts while the relations found in an `Objective` allow for the quantification of the learning concept and providing paths to self-learning.

The Topology OSID may also be used to construct and view a concept map. While a Topology OSID Provider may be adapted from a Learning OSID or an Ontology OSID, the topology for either would be interpreted from a multi-parented hierarchy of the `Objectives` and `Subjects` respectively.

Courses

The Learning OSID may be used in conjunction with the Course OSID to identify dsired learning oitcomes from a course or to align the course activities and syllabus with stated learning objectives. The Course OSID describes learning from a structured curriculum management point of view where the Learning OSID and allows for various objectives to be combined and related without any regard to a prescribed curriculum.

Sub Packages

The Learning OSID contains a Learning Batch OSID for bulk management of `Objectives, Activities,` and `Proficiencies` . Learning Open Service Interface Definitions learning version 3.0.0

The Learning OSID manages learning objectives. A learning `Objective` describes measurable learning goals.

Objectives

`Objectives` describe measurable learning goals. A learning objective may be measured by a related `Assesment`. `Objectives` may be mapped to levels, A level is represented by a `Grade` which is used to indicate a grade level or level of difficulty.

`Objectives` are hierarchical. An `Objective` with children represents an objective that is inclusive of all its children. For example, an `Objective` that represents learning in arithmetic may be composed of objectives that represent learning in both addition and subtraction.

`Objectives` may also have requisites. A requisite objective is one that should be achieved before an objective is attempted.

Activities

An `Activity` describes actions that one can do to meet a learning objective. An `Activity` includes a list of `Assets` to read or watch, or a list of `Courses` to take, or a list of learning `Assessments` to practice. An `Activity` may also represent other learning activities such as taking a course or practicing an instrument. An `Activity` is specific to an `Objective` where the reusability is achieved based on what the `Activity` relates.

Proficiencies

A `Proficiency` is an `OsidRelationship` measuring the competence of a `Resource` with respect to an Objective.

Objective Bank Cataloging

`Objectives, Activities,` and `Proficiencies` can be organized into hierarchical `ObjectiveBanks` for the purposes of categorization and federation.

Concept Mapping

A concept can be modeled as a learning `Objective` without any related `Assessment` or `Activities`. In this scenario, an `Objective` looks much like the simpler `Subject` in the Ontology OSID. The Ontology OSID is constrained to qualifying concepts while the relations found in an `Objective` allow for the quantification of the learning concept and providing paths to self-learning.

The Topology OSID may also be used to construct and view a concept map. While a Topology OSID Provider may be adapted from a Learning OSID or an Ontology OSID, the topology for either would be interpreted from a multi-parented hierarchy of the `Objectives` and `Subjects` respectively.

Courses

The Learning OSID may be used in conjunction with the Course OSID to identify dsired learning oitcomes from a course or to align the course activities and syllabus with stated learning objectives. The Course OSID describes learning from a structured curriculum management point of view where the Learning OSID and allows for various objectives to be combined and related without any regard to a prescribed curriculum.

Sub Packages

The Learning OSID contains a Learning Batch OSID for bulk management of `Objectives, Activities, and Proficiencies.`

Service Managers

Learning Profile

class `dlkit.services.learning.`**`LearningProfile`**

Bases: `dlkit.osid.managers.OsidProfile`

The `LearningProfile` describes the interoperability among learning services.

`supports_objective_lookup``()`

Tests if an objective lookup service is supported.

An objective lookup service defines methods to access objectives.

Returns true if objective lookup is supported, false otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_objective_query``()`

Tests if an objective query service is supported.

Returns `true` if objective query is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_objective_admin``()`

Tests if an objective administrative service is supported.

Returns `true` if objective admin is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_objective_hierarchy``()`
Tests if an objective hierarchy traversal is supported.

Returns `true` if an objective hierarchy traversal is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_objective_hierarchy_design``()`
Tests if an objective hierarchy design is supported.

Returns `true` if an objective hierarchy design is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_objective_sequencing``()`
Tests if an objective sequencing design is supported.

Returns `true` if objective sequencing is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_objective_objective_bank``()`
Tests if an objective to objective bank lookup session is available.

Returns `true` if objective objective bank lookup session is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_objective_objective_bank_assignment``()`
Tests if an objective to objective bank assignment session is available.

Returns `true` if objective objective bank assignment is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_objective_requisite``()`
Tests if an objective requisite service is supported.

Returns `true` if objective requisite service is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_objective_requisite_assignment``()`
Tests if an objective requisite assignment service is supported.

Returns `true` if objective requisite assignment service is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_activity_lookup()
Tests if an activity lookup service is supported.

Returns `true` if activity lookup is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_activity_admin()
Tests if an activity administrative service is supported.

Returns `true` if activity admin is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_activity_objective_bank()
Tests if an activity to objective bank lookup session is available.

Returns `true` if activity objective bank lookup session is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_activity_objective_bank_assignment()
Tests if an activity to objective bank assignment session is available.

Returns `true` if activity objective bank assignment is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_proficiency_lookup()
Tests if looking up proficiencies is supported.

Returns `true` if proficiency lookup is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_proficiency_query()
Tests if querying proficiencies is supported.

Returns `true` if proficiency query is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_proficiency_admin()
Tests if proficiencyadministrative service is supported.

Returns `true` if proficiency administration is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_objective_bank_lookup()
Tests if an objective bank lookup service is supported.

Returns `true` if objective bank lookup is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_objective_bank_admin()

Tests if an objective bank administrative service is supported.

Returns `true` if objective bank admin is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_objective_bank_hierarchy()

Tests if an objective bank hierarchy traversal is supported.

Returns `true` if an objective bank hierarchy traversal is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_objective_bank_hierarchy_design()

Tests if objective bank hierarchy design is supported.

Returns `true` if an objective bank hierarchy design is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

objective_record_types

Gets the supported `Objective` record types.

Returns a list containing the supported `Objective` record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

objective_search_record_types

Gets the supported `Objective` search record types.

Returns a list containing the supported `Objective` search record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

activity_record_types

Gets the supported `Activity` record types.

Returns a list containing the supported `Activity` record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

activity_search_record_types

Gets the supported `Activity` search record types.

Returns a list containing the supported `Activity` search record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

proficiency_record_types

Gets the supported `Proficiency` record types.

Returns a list containing the supported `Proficiency` record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

`proficiency_search_record_types`
Gets the supported `Proficiency` search types.

Returns a list containing the supported `Proficiency` search types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

`objective_bank_record_types`
Gets the supported `ObjectiveBank` record types.

Returns a list containing the supported `ObjectiveBank` record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

`objective_bank_search_record_types`
Gets the supported objective bank search record types.

Returns a list containing the supported `ObjectiveBank` search record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

Learning Manager

class `dlkit.services.learning.`**`LearningManager`**(*proxy=None*)
Bases: `dlkit.osid.managers.OsidManager`, `dlkit.osid.sessions.OsidSession`, *`dlkit.services.learning.LearningProfile`*

The learning manager provides access to learning sessions and provides interoperability tests for various aspects of this service.

The sessions included in this manager are:

- `ObjectiveLookupSession:` a session to look up objectives
- `ObjectiveLookupSession:` a session to query objectives `None`
- `ObjectiveSearchSession:` a session to search objectives
- `ObjectiveAdminSession:` a session to create, modify and delete objectives `None`
- `ObjectiveNotificationSession:` a session to receive messages pertaining to objective "" changes
- `ObjectiveHierarchySession:` a session to traverse objective hierarchies
- `ObjectiveHierarchyDesignSession:` a session to design objective hierarchies
- `ObjectiveSequencingSession:` a session to sequence objectives
- `ObjectiveObjectiveBankSession:` a session for retriieving objective and objective bank mappings
- `ObjectiveObjectiveBankAssignmentSession:` a session for managing objective and objective bank mappings
- `ObjectiveSmartObjectiveBankSession:` a session for managing dynamic objective banks

- `ObjectiveRequisiteSession`: a session to examine objective requisites
- `ObjectiveRequisiteAssignmentSession`: a session to manage objective requisites
- `ActivityLookupSession`: a session to look up activities
- `ActivityQuerySession`: a session to query activities `None`
- `ActivitySearchSession`: a session to search activities
- `ActivityAdminSession`: a session to create, modify and delete activities `None`
- `ActivityNotificationSession`: a session to receive messages pertaining to activity ``` changes
- `ActivityObjectiveBankSession`: a session for retriieving activity and objective bank mappings
- `ActivityObjectiveBankAssignmentSession`: a session for managing activity and objective bank mappings
- `ActivitySmartObjectiveBankSession`: a session for managing dynamic objective banks of activities
- `ProficiencyLookupSession`: a session to retrieve proficiencies
- `ProficiencyQuerySession`: a session to query proficiencies
- `ProficiencySearchSession`: a session to search for proficiencies
- `ProficiencyAdminSession`: a session to create, update, and delete proficiencies
- `ProficiencyNotificationSession`: a session to receive notifications pertaining to proficiency changes
- `ProficiencyObjectiveBankSession`: a session to look up proficiency to objective bank mappings
- `ProficiencyObjectiveBankAssignmentSession`: a session to manage proficiency to objective bank mappings
- `ProficiencySmartObjectiveBankSession`: a session to manage smart objective banks of proficiencies
- `MyLearningPathSession`: a session to examine learning paths of objectives
- `LearningPathSession`: a session to examine learning paths of objectives
- `ObjectiveBankLookupSession`: a session to lookup objective banks
- `ObjectiveBankQuerySession`: a session to query objective banks
- `ObjectiveBankSearchSession` : a session to search objective banks
- `ObjectiveBankAdminSession` : a session to create, modify and delete objective banks
- `ObjectiveBankNotificationSession` : a session to receive messages pertaining to objective bank changes
- `ObjectiveBankHierarchySession`: a session to traverse the objective bank hierarchy
- `ObjectiveBankHierarchyDesignSession`: a session to manage the objective bank hierarchy

`learning_batch_manager`

Gets a `LearningBatchManager`.

Returns a `LearningBatchManager`

Return type `osid.learning.batch.LearningBatchManager`

Raise `OperationFailed` – unable to complete request

Raise `Unimplemented` – `supports_learning_batch() is false`

compliance: optional – This method must be implemented if ``supports_learning_batch()`` is true.

Objective Bank Lookup Methods

`LearningManager.`**`can_lookup_objective_banks`**`()`

Tests if this user can perform `ObjectiveBank` lookups.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if lookup methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`LearningManager.`**`use_comparative_objective_bank_view`**`()`

The returns from the objective bank methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`LearningManager.`**`use_plenary_objective_bank_view`**`()`

A complete view of the `Hierarchy` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`LearningManager.`**`get_objective_bank`**`(`*objective_bank_id*`)`

Gets the `ObjectiveBank` specified by its `Id`.

In plenary mode, the exact `Id` is found or a `NotFound` results. Otherwise, the returned `ObjectiveBank` may have a different `Id` than requested, such as the case where a duplicate `Id` was assigned to a `ObjectiveBank` and retained for compatility.

Parameters **`objective_bank_id`** (`osid.id.Id`) – `Id` of the `ObjectiveBank`

Returns the objective bank

Return type `osid.learning.ObjectiveBank`

Raise `NotFound` – `objective_bank_id` not found

Raise `NullArgument` – `objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method is must be implemented.

`LearningManager.`**`get_objective_banks_by_ids`**`(`*objective_bank_ids*`)`

Gets a `ObjectiveBankList` corresponding to the given `IdList`.

In plenary mode, the returned list contains all of the objective banks specified in the `Id` list, in the order of the list, including duplicates, or an error results if an `Id` in the supplied list is not found or inaccessible. Otherwise, inaccessible `ObjectiveBank` objects may be omitted from the list and may present the elements in any order including returning a unique set.

Parameters **`objective_bank_ids`** (`osid.id.IdList`) – the list of `Ids` to retrieve

Returns the returned `ObjectiveBank` list

Return type `osid.learning.ObjectiveBankList`

Raise `NotFound` – an `Id was` not found

Raise `NullArgument` – `objective_bank_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`LearningManager.`**`get_objective_banks_by_genus_type`**(*objective_bank_genus_type*)

Gets a `ObjectiveBankList` corresponding to the given objective bank genus `Type` which does not include objective banks of types derived from the specified `Type`.

In plenary mode, the returned list contains all known objective banks or an error results. Otherwise, the returned list may contain only those objective banks that are accessible through this session.

Parameters **`objective_bank_genus_type`** (`osid.type.Type`) – an objective bank genus type

Returns the returned `ObjectiveBank` list

Return type `osid.learning.ObjectiveBankList`

Raise `NullArgument` – `objective_bank_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`LearningManager.`**`get_objective_banks_by_parent_genus_type`**(*objective_bank_genus_type*)

Gets a `ObjectiveBankList` corresponding to the given objective bank genus `Type` and include any additional objective banks with genus types derived from the specified `Type`.

In plenary mode, the returned list contains all known objective banks or an error results. Otherwise, the returned list may contain only those objective banks that are accessible through this session.

Parameters **`objective_bank_genus_type`** (`osid.type.Type`) – an objective bank genus type

Returns the returned `ObjectiveBank` list

Return type `osid.learning.ObjectiveBankList`

Raise `NullArgument` – `objective_bank_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`LearningManager.`**`get_objective_banks_by_record_type`**(*objective_bank_record_type*)

Gets a `ObjectiveBankList` containing the given objective bank record `Type`.

In plenary mode, the returned list contains all known objective banks or an error results. Otherwise, the returned list may contain only those objective banks that are accessible through this session.

Parameters **`objective_bank_record_type`** (`osid.type.Type`) – an objective bank record type

Returns the returned `ObjectiveBank` list

Return type `osid.learning.ObjectiveBankList`

Raise `NullArgument` – `objective_bank_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`LearningManager.`**`get_objective_banks_by_provider`**(*resource_id*)

Gets a `ObjectiveBankList` for the given provider.

In plenary mode, the returned list contains all known objective banks or an error results. Otherwise, the returned list may contain only those objective banks that are accessible through this session.

Parameters **`resource_id`** (`osid.id.Id`) – a resource `Id`

Returns the returned `ObjectiveBank` list

Return type `osid.learning.ObjectiveBankList`

Raise `NullArgument` – `resource_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`LearningManager.`**`objective_banks`**

Gets all `ObjectiveBanks`.

In plenary mode, the returned list contains all known objective banks or an error results. Otherwise, the returned list may contain only those objective banks that are accessible through this session.

Returns a `ObjectiveBankList`

Return type `osid.learning.ObjectiveBankList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Objective Bank Admin Methods

`LearningManager.`**`can_create_objective_banks`**()

Tests if this user can create `ObjectiveBanks`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known creating an `ObjectiveBank` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer create operations to unauthorized users.

Returns `false` if `ObjectiveBank` creation is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`LearningManager.`**`can_create_objective_bank_with_record_types`**`(`*objective_bank_record_types*`)`

Tests if this user can create a single `ObjectiveBank` using the desired record types.

While `LearningManager.getObjectiveBankRecordTypes()` can be used to examine which records are supported, this method tests which record(s) are required for creating a specific `ObjectiveBank`. Providing an empty array tests if an `ObjectiveBank` can be created with no records.

Parameters **`objective_bank_record_types`** (`osid.type.Type[]`) – array of objective bank record types

Returns `true` if `ObjectiveBank` creation using the specified `Types` is supported, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `objective_bank_record_types` is `null`

compliance: mandatory – This method must be implemented.

`LearningManager.`**`get_objective_bank_form_for_create`**`(`*objective_bank_record_types*`)`

Gets the objective bank form for creating new objective banks.

A new form should be requested for each create transaction.

Parameters **`objective_bank_record_types`** (`osid.type.Type[]`) – array of objective bank record types

Returns the objective bank form

Return type `osid.learning.ObjectiveBankForm`

Raise `NullArgument` – `objective_bank_record_types` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – unable to get form for requested record types.

compliance: mandatory – This method must be implemented.

`LearningManager.`**`create_objective_bank`**`(`*objective_bank_form*`)`

Creates a new `ObjectiveBank`.

Parameters **`objective_bank_form`** (`osid.learning.ObjectiveBankForm`) – the form for this `ObjectiveBank`

Returns the new `ObjectiveBank`

Return type `osid.learning.ObjectiveBank`

Raise `IllegalState` – `objective_bank_form` already used in a create transaction

Raise `InvalidArgument` – one or more of the form elements is invalid

Raise `NullArgument` – `objective_bank_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `objective_bank_form` did not originate from `get_objective_bank_form_for_create()`

compliance: mandatory – This method must be implemented.

`LearningManager.`**`can_update_objective_banks`**`()`

Tests if this user can update `ObjectiveBanks`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating an `ObjectiveBank` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer update operations to unauthorized users.

Returns `false` if `ObjectiveBank` modification is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`LearningManager.`**`get_objective_bank_form_for_update`**`(`*objective_bank_id*`)`

Gets the objective bank form for updating an existing objective bank.

A new objective bank form should be requested for each update transaction.

Parameters **`objective_bank_id`** (`osid.id.Id`) – the `Id` of the `ObjectiveBank`

Returns the objective bank form

Return type `osid.learning.ObjectiveBankForm`

Raise `NotFound` – `objective_bank_id` is not found

Raise `NullArgument` – `objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`LearningManager.`**`update_objective_bank`**`(`*objective_bank_form*`)`

Updates an existing objective bank.

Parameters **`objective_bank_form`** (`osid.learning.ObjectiveBankForm`) – the form containing the elements to be updated

Raise `IllegalState` – `objective_bank_form` already used in an update transaction

Raise `InvalidArgument` – the form contains an invalid value

Raise `NullArgument` – `objective_bank_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `objective_bank_form did not originate from get_objective_bank_form_for_update()`

compliance: mandatory – This method must be implemented.

`LearningManager.`**`can_delete_objective_banks`**`()`

Tests if this user can delete objective banks.

A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting an `ObjectiveBank` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer delete operations to unauthorized users.

Returns `false` if `ObjectiveBank` deletion is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`LearningManager.`**`delete_objective_bank`**`(`*objective_bank_id*`)`

Deletes an `ObjectiveBank`.

Parameters **`objective_bank_id`** (`osid.id.Id`) – the `Id` of the `ObjectiveBank` to remove

Raise `NotFound` – `objective_bank_id` not found

Raise `NullArgument` – `objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`LearningManager.`**`can_manage_objective_bank_aliases`**`()`

Tests if this user can manage `Id` aliases for `ObjectiveBanks`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known changing an alias will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer alias operations to an unauthorized user.

Returns `false` if `ObjectiveBank` aliasing is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`LearningManager.`**`alias_objective_bank`**`(`*objective_bank_id*, *alias_id*`)`

Adds an `Id` to an `ObjectiveBank` for the purpose of creating compatibility.

The primary `Id` of the `ObjectiveBank` is determined by the provider. The new `Id` performs as an alias to the primary `Id`. If the alias is a pointer to another objective bank, it is reassigned to the given objective bank `Id`.

Parameters

- **`objective_bank_id`** (`osid.id.Id`) – the `Id` of an `ObjectiveBank`
- **`alias_id`** (`osid.id.Id`) – the alias `Id`

Raise `AlreadyExists` – `alias_id` is already assigned

Raise `NotFound` – `objective_bank_id` not found

Raise `NullArgument` – `objective_bank_id` or `alias_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Objective Bank Hierarchy Methods

`LearningManager.`**`objective_bank_hierarchy_id`**

Gets the hierarchy `Id` associated with this session.

Returns the hierarchy `Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`LearningManager.`**`objective_bank_hierarchy`**

Gets the hierarchy associated with this session.

Returns the hierarchy associated with this session

Return type `osid.hierarchy.Hierarchy`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`LearningManager.`**`can_access_objective_bank_hierarchy`**`()`

Tests if this user can perform hierarchy queries.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an an application that may not offer traversal functions to unauthorized users.

Returns `false` if hierarchy traversal methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`LearningManager.`**`use_comparative_objective_bank_view`**`()`

The returns from the objective bank methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`LearningManager.`**`use_plenary_objective_bank_view`**`()`

A complete view of the `Hierarchy` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`LearningManager.`**`root_objective_bank_ids`**

Gets the root objective bank `Ids` in this hierarchy.

Returns the root objective bank `Ids`

Return type `osid.id.IdList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`LearningManager.`**`root_objective_banks`**

Gets the root objective banks in this objective bank hierarchy.

Returns the root objective banks

Return type `osid.learning.ObjectiveBankList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method is must be implemented.

`LearningManager.`**`has_parent_objective_banks`**`(`*objective_bank_id*`)`

Tests if the `ObjectiveBank` has any parents.

Parameters **`objective_bank_id`** (`osid.id.Id`) – the `Id` of an objective bank

Returns `true` if the objective bank has parents, `false` otherwise

Return type `boolean`

Raise `NotFound` – `objective_bank_id` is not found

Raise `NullArgument` – `objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`LearningManager.`**`is_parent_of_objective_bank`**`(`*id_*, *objective_bank_id*`)`

Tests if an `Id` is a direct parent of an objective bank.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`objective_bank_id`** (`osid.id.Id`) – the `Id` of an objective bank

Returns `true` if this `id` is a parent of `objective_bank_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `objective_bank_id` is not found

Raise `NullArgument` – `id` or `objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` not found return `false`.

`LearningManager.`**`get_parent_objective_bank_ids`**`(`*objective_bank_id*`)`

Gets the parent `Ids` of the given objective bank.

Parameters **`objective_bank_id`** (`osid.id.Id`) – the `Id` of an objective bank

Returns the parent `Ids` of the objective bank

Return type `osid.id.IdList`

Raise `NotFound` – `objective_bank_id` is not found

Raise `NullArgument` – `objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`LearningManager.`**`get_parent_objective_banks`**`(`*objective_bank_id*`)`

Gets the parents of the given objective bank.

Parameters **`objective_bank_id`** (`osid.id.Id`) – the `Id` of an objective bank

Returns the parents of the objective bank

Return type `osid.learning.ObjectiveBankList`

Raise `NotFound` – `objective_bank_id` is not found

Raise `NullArgument` – `objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`LearningManager.`**`is_ancestor_of_objective_bank`**(*id_*, *objective_bank_id*)

Tests if an `Id` is an ancestor of an objective bank.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`objective_bank_id`** (`osid.id.Id`) – the `Id` of an objective bank

Returns `true` if this `id` is an ancestor of `objective_bank_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `objective_bank_id` is not found

Raise `NullArgument` – `id` or `objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` not found return `false`.

`LearningManager.`**`has_child_objective_banks`**(*objective_bank_id*)

Tests if an objective bank has any children.

Parameters **`objective_bank_id`** (`osid.id.Id`) – the `Id` of an objective bank

Returns `true` if the `objective_bank_id` has children, `false` otherwise

Return type `boolean`

Raise `NotFound` – `objective_bank_id` is not found

Raise `NullArgument` – `objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`LearningManager.`**`is_child_of_objective_bank`**(*id_*, *objective_bank_id*)

Tests if an objective bank is a direct child of another.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`objective_bank_id`** (`osid.id.Id`) – the `Id` of an objective bank

Returns `true` if the `id` is a child of `objective_bank_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `objective_bank_id` is not found

Raise `NullArgument` – `id` or `objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` not found return `false`.

`LearningManager.`**`get_child_objective_bank_ids`**`(`*objective_bank_id*`)`

Gets the child `Ids` of the given objective bank.

Parameters **`objective_bank_id`** (`osid.id.Id`) – the `Id` to query

Returns the children of the objective bank

Return type `osid.id.IdList`

Raise `NotFound` – `objective_bank_id` is not found

Raise `NullArgument` – `objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`LearningManager.`**`get_child_objective_banks`**`(`*objective_bank_id*`)`

Gets the children of the given objective bank.

Parameters **`objective_bank_id`** (`osid.id.Id`) – the `Id` to query

Returns the children of the objective bank

Return type `osid.learning.ObjectiveBankList`

Raise `NotFound` – `objective_bank_id` is not found

Raise `NullArgument` – `objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`LearningManager.`**`is_descendant_of_objective_bank`**`(`*id_*, *objective_bank_id*`)`

Tests if an `Id` is a descendant of an objective bank.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`objective_bank_id`** (`osid.id.Id`) – the `Id` of an objective bank

Returns `true` if the `id` is a descendant of the `objective_bank_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `objective_bank_id` is not found

Raise `NullArgument` – `id` or `objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` is not found return `false`.

`LearningManager.`**`get_objective_bank_node_ids`**(*objective_bank_id*, *ancestor_levels*, *descendant_levels*, *include_siblings*)

Gets a portion of the hierarchy for the given objective bank.

Parameters

- **`objective_bank_id`** (`osid.id.Id`) – the `Id` to query
- **`ancestor_levels`** (`cardinal`) – the maximum number of ancestor levels to include. A value of 0 returns no parents in the node.
- **`descendant_levels`** (`cardinal`) – the maximum number of descendant levels to include. A value of 0 returns no children in the node.
- **`include_siblings`** (`boolean`) – `true` to include the siblings of the given node, `false` to omit the siblings

Returns a catalog node

Return type `osid.hierarchy.Node`

Raise `NotFound` – `objective_bank_id` not found

Raise `NullArgument` – `objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`LearningManager.`**`get_objective_bank_nodes`**(*objective_bank_id*, *ancestor_levels*, *descendant_levels*, *include_siblings*)

Gets a portion of the hierarchy for the given objective bank.

Parameters

- **`objective_bank_id`** (`osid.id.Id`) – the `Id` to query
- **`ancestor_levels`** (`cardinal`) – the maximum number of ancestor levels to include. A value of 0 returns no parents in the node.
- **`descendant_levels`** (`cardinal`) – the maximum number of descendant levels to include. A value of 0 returns no children in the node.
- **`include_siblings`** (`boolean`) – `true` to include the siblings of the given node, `false` to omit the siblings

Returns an objective bank node

Return type `osid.learning.ObjectiveBankNode`

Raise `NotFound` – `objective_bank_id` not found

Raise `NullArgument` – `objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Objective Bank Hierarchy Design Methods

`LearningManager.`**`objective_bank_hierarchy_id`**

Gets the hierarchy `Id` associated with this session.

Returns the hierarchy `Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`LearningManager.`**`objective_bank_hierarchy`**

Gets the hierarchy associated with this session.

Returns the hierarchy associated with this session

Return type `osid.hierarchy.Hierarchy`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`LearningManager.`**`can_modify_objective_bank_hierarchy`**`()`

Tests if this user can change the hierarchy.

A return of true does not guarantee successful authorization. A return of false indicates that it is known performing any update will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer these operations to an unauthorized user.

Returns `false` if changing this hierarchy is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`LearningManager.`**`add_root_objective_bank`**`(`*objective_bank_id*`)`

Adds a root objective bank.

Parameters **`objective_bank_id`** (`osid.id.Id`) – the `Id` of an objective bank

Raise `AlreadyExists` – `objective_bank_id` is already in hierarchy

Raise `NotFound` – `objective_bank_id` not found

Raise `NullArgument` – `objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`LearningManager.`**`remove_root_objective_bank`**`(`*objective_bank_id*`)`

Removes a root objective bank.

Parameters **`objective_bank_id`** (`osid.id.Id`) – the `Id` of an objective bank

Raise `NotFound` – `objective_bank_id` is not a root

Raise `NullArgument` – `objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`LearningManager.`**`add_child_objective_bank`**(*objective_bank_id*, *child_id*)

Adds a child to an objective bank.

Parameters

- **`objective_bank_id`** (`osid.id.Id`) – the `Id` of an objective bank
- **`child_id`** (`osid.id.Id`) – the `Id` of the new child

Raise `AlreadyExists` – `objective_bank_id` is already a parent of `child_id`

Raise `NotFound` – `objective_bank_id` or `child_id` not found

Raise `NullArgument` – `objective_bank_id` or `child_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`LearningManager.`**`remove_child_objective_bank`**(*objective_bank_id*, *child_id*)

Removes a child from an objective bank.

Parameters

- **`objective_bank_id`** (`osid.id.Id`) – the `Id` of an objective bank
- **`child_id`** (`osid.id.Id`) – the `Id` of the child

Raise `NotFound` – `objective_bank_id` not a parent of `child_id`

Raise `NullArgument` – `objective_bank_id` or `child_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`LearningManager.`**`remove_child_objective_banks`**(*objective_bank_id*)

Removes all children from an objective bank.

Parameters **`objective_bank_id`** (`osid.id.Id`) – the `Id` of an objective bank

Raise `NotFound` – `objective_bank_id` not in hierarchy

Raise `NullArgument` – `objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Objective Bank

Objective Bank

class `dlkit.services.learning.`**`ObjectiveBank`**(*provider_manager*, *catalog*, *runtime*, *proxy*, ***kwargs*)

Bases: *`dlkit.osid.objects.OsidCatalog`*, `dlkit.osid.sessions.OsidSession`

an objective bank defines a collection of objectives.

get_objective_bank_record(*objective_bank_record_type*)
Gets the objective bank record corresponding to the given `ObjectiveBank` record `Type`.

This method is used to retrieve an object implementing the requested record. The `objective_bank_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(objective_bank_record_type)` is `true`.

Parameters **objective_bank_record_type** (`osid.type.Type`) – an objective bank record type

Returns the objective bank record

Return type `osid.learning.records.ObjectiveBankRecord`

Raise `NullArgument` – `objective_bank_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(objective_bank_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Objective Lookup Methods

`ObjectiveBank.`**objective_bank_id**
Gets the `ObjectiveBank Id` associated with this session.

Returns the `ObjectiveBank Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**objective_bank**
Gets the `ObjectiveBank` associated with this session.

Returns the obective bank

Return type `osid.learning.ObjectiveBank`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**can_lookup_objectives**()
Tests if this user can perform `Objective` lookups.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if lookup methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**use_comparative_objective_view**()
The returns from the lookup methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`use_plenary_objective_view`**`()`

A complete view of the `Objective` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`use_federated_objective_bank_view`**`()`

Federates the view for methods in this session.

A federated view will include proficiencies in objective banks which are children of this objective bank in the obective bank hierarchy.

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`use_isolated_objective_bank_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts lookups to this objective bank only.

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`get_objective`**`(`*objective_id*`)`

Gets the `Objective` specified by its `Id`.

In plenary mode, the exact `Id` is found or a `NotFound` results. Otherwise, the returned `Objective` may have a different `Id` than requested, such as the case where a duplicate `Id` was assigned to an `Objective` and retained for compatibility.

Parameters **`objective_id`** (`osid.id.Id`) – `Id` of the `Objective`

Returns the objective

Return type `osid.learning.Objective`

Raise `NotFound` – `objective_id` not found

Raise `NullArgument` – `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`get_objectives_by_ids`**`(`*objective_ids*`)`

Gets an `ObjectiveList` corresponding to the given `IdList`.

In plenary mode, the returned list contains all of the objectives specified in the `Id` list, in the order of the list, including duplicates, or an error results if an `Id` in the supplied list is not found or inaccessible. Otherwise, inaccessible `Objectives` may be omitted from the list and may present the elements in any order including returning a unique set.

Parameters **`objective_ids`** (`osid.id.IdList`) – the list of `Ids` to retrieve

Returns the returned `Objective` list

Return type `osid.learning.ObjectiveList`

Raise `NotFound` – an `Id was` not found

Raise `NullArgument` – `objective_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_objectives_by_genus_type`**(*objective_genus_type*)

Gets an `ObjectiveList` corresponding to the given objective genus `Type` which does not include objectives of genus types derived from the specified `Type`.

In plenary mode, the returned list contains all known objectives or an error results. Otherwise, the returned list may contain only those objectives that are accessible through this session.

Parameters **`objective_genus_type`** (`osid.type.Type`) – an objective genus type

Returns the returned `Objective` list

Return type `osid.learning.ObjectiveList`

Raise `NullArgument` – `objective_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_objectives_by_parent_genus_type`**(*objective_genus_type*)

Gets an `ObjectiveList` corresponding to the given objective genus `Type` and include any additional objective with genus types derived from the specified `Type`.

In plenary mode, the returned list contains all known objectives or an error results. Otherwise, the returned list may contain only those objectives that are accessible through this session

Parameters **`objective_genus_type`** (`osid.type.Type`) – an objective genus type

Returns the returned `Objective` list

Return type `osid.learning.ObjectiveList`

Raise `NullArgument` – `objective_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_objectives_by_record_type`**(*objective_record_type*)

Gets an `ObjectiveList` containing the given objective record `Type`.

In plenary mode, the returned list contains all known objectives or an error results. Otherwise, the returned list may contain only those objectives that are accessible through this session.

Parameters **`objective_record_type`** (`osid.type.Type`) – an objective record type

Returns the returned `Objective` list

Return type `osid.learning.ObjectiveList`

Raise `NullArgument` – `objective_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`objectives`**

Gets all `Objectives`.

In plenary mode, the returned list contains all known objectives or an error results. Otherwise, the returned list may contain only those objectives that are accessible through this session.

Returns an `ObjectiveList`

Return type `osid.learning.ObjectiveList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Objective Query Methods

`ObjectiveBank.`**`objective_bank_id`**

Gets the `ObjectiveBank Id` associated with this session.

Returns the `ObjectiveBank Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`objective_bank`**

Gets the `ObjectiveBank` associated with this session.

Returns the obective bank

Return type `osid.learning.ObjectiveBank`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`can_search_objectives`**`()`

Tests if this user can perform `Objectives` searches.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer search operations to unauthorized users.

Returns `false` if search methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`use_federated_objective_bank_view`**`()`

Federates the view for methods in this session.

A federated view will include proficiencies in objective banks which are children of this objective bank in the obective bank hierarchy.

compliance: mandatory – This method is must be implemented.

ObjectiveBank.**use_isolated_objective_bank_view**()

Isolates the view for methods in this session.

An isolated view restricts lookups to this objective bank only.

compliance: mandatory – This method is must be implemented.

ObjectiveBank.**objective_query**

Gets an objective query.

Returns the objective query

Return type `osid.learning.ObjectiveQuery`

compliance: mandatory – This method must be implemented.

ObjectiveBank.**get_objectives_by_query**(*objective_query*)

Gets a list of `Objectives` matching the given objective query.

Parameters **objective_query** (`osid.learning.ObjectiveQuery`) – the objective query

Returns the returned `ObjectiveList`

Return type `osid.learning.ObjectiveList`

Raise `NullArgument` – `objective_query` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `objective_query` is not of this service

compliance: mandatory – This method must be implemented.

Objective Admin Methods

ObjectiveBank.**objective_bank_id**

Gets the `ObjectiveBank Id` associated with this session.

Returns the `ObjectiveBank Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

ObjectiveBank.**objective_bank**

Gets the `ObjectiveBank` associated with this session.

Returns the obective bank

Return type `osid.learning.ObjectiveBank`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

ObjectiveBank.**can_create_objectives**()

Tests if this user can create `Objectives`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known creating an Objective will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer create operations to an unauthorized user.

Returns `false` if `Objective` creation is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`can_create_objective_with_record_types`**`(`*objective_record_types*`)`

Tests if this user can create a single `Objective` using the desired record types.

While `LearningManager.getObjectiveRecordTypes()` can be used to examine which records are supported, this method tests which record(s) are required for creating a specific `Objective`. Providing an empty array tests if an `Objective` can be created with no records.

Parameters **`objective_record_types`** (`osid.type.Type[]`) – array of objective record types

Returns `true` if `Objective` creation using the specified record `Types` is supported, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `objective_record_types` is `null`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_objective_form_for_create`**`(`*objective_record_types*`)`

Gets the objective form for creating new objectives.

A new form should be requested for each create transaction.

Parameters **`objective_record_types`** (`osid.type.Type[]`) – array of objective record types

Returns the objective form

Return type `osid.learning.ObjectiveForm`

Raise `NullArgument` – `objective_record_types` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – unable to get form for requested record types

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`create_objective`**`(`*objective_form*`)`

Creates a new `Objective`.

Parameters **`objective_form`** (`osid.learning.ObjectiveForm`) – the form for this `Objective`

Returns the new `Objective`

Return type `osid.learning.Objective`

Raise `IllegalState` – `objective_form` already used in a create transaction

Raise `InvalidArgument` – one or more of the form elements is invalid

Raise `NullArgument` – `objective_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `objective_form` did not originate from `get_objective_form_for_create()`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`can_update_objectives`**`()`

Tests if this user can update `Objectives`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating an `Objective` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer update operations to an unauthorized user.

Returns `false` if objective modification is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_objective_form_for_update`**`(`*objective_id*`)`

Gets the objective form for updating an existing objective.

A new objective form should be requested for each update transaction.

Parameters **`objective_id`** (`osid.id.Id`) – the `Id` of the `Objective`

Returns the objective form

Return type `osid.learning.ObjectiveForm`

Raise `NotFound` – `objective_id` is not found

Raise `NullArgument` – `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`update_objective`**`(`*objective_form*`)`

Updates an existing objective.

Parameters **`objective_form`** (`osid.learning.ObjectiveForm`) – the form containing the elements to be updated

Raise `IllegalState` – `objective_form` already used in an update transaction

Raise `InvalidArgument` – the form contains an invalid value

Raise `NullArgument` – `objective_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `objective_form` did not originate from `get_objective_form_for_update()`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`can_delete_objectives`**`()`

Tests if this user can delete `Objectives`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting an `Objective` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer delete operations to an unauthorized user.

Returns `false` if `Objective` deletion is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`delete_objective`**`(`*objective_id*`)`

Deletes the `Objective` identified by the given `Id`.

Parameters **`objective_id`** (`osid.id.Id`) – the `Id` of the `Objective` to delete

Raise `NotFound` – an `Objective` was not found identified by the given `Id`

Raise `NullArgument` – `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`can_manage_objective_aliases`**`()`

Tests if this user can manage `Id` aliases for `Objectives`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known changing an alias will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer alias operations to an unauthorized user.

Returns `false` if `Objective` aliasing is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`alias_objective`**`(`*objective_id*, *alias_id*`)`

Adds an `Id` to an `Objective` for the purpose of creating compatibility.

The primary `Id` of the `Objective` is determined by the provider. The new `Id` performs as an alias to the primary `Id`. If the alias is a pointer to another objective, it is reassigned to the given objective `Id`.

Parameters

- **`objective_id`** (`osid.id.Id`) – the `Id` of an `Objective`
- **`alias_id`** (`osid.id.Id`) – the alias `Id`

Raise `AlreadyExists` – `alias_id` is already assigned

Raise `NotFound` – `objective_id` not found

Raise `NullArgument` – `objective_id` or `alias_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Objective Hierarchy Methods

`ObjectiveBank.`**`objective_hierarchy_id`**

Gets the hierarchy `Id` associated with this session.

Returns the hierarchy `Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`objective_hierarchy`**

Gets the hierarchy associated with this session.

Returns the hierarchy associated with this session

Return type `osid.hierarchy.Hierarchy`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`can_access_objective_hierarchy`**`()`

Tests if this user can perform hierarchy queries.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an an application that may not offer traversal functions to unauthorized users.

Returns `false` if hierarchy traversal methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`use_comparative_objective_view`**`()`

The returns from the lookup methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`use_plenary_objective_view`**`()`

A complete view of the `Objective` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`root_objective_ids`**

Gets the root objective `Ids` in this hierarchy.

Returns the root objective `Ids`

Return type `osid.id.IdList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`root_objectives`**

Gets the root objective in this objective hierarchy.

Returns the root objective

Return type `osid.learning.ObjectiveList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`has_parent_objectives`**(*objective_id*)

Tests if the `Objective` has any parents.

Parameters **`objective_id`** (`osid.id.Id`) – the `Id` of an objective

Returns `true` if the objective has parents, `false` otherwise

Return type `boolean`

Raise `NotFound` – `objective_id` is not found

Raise `NullArgument` – `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`is_parent_of_objective`**(*id_*, *objective_id*)

Tests if an `Id` is a direct parent of an objective.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`objective_id`** (`osid.id.Id`) – the `Id` of an objective

Returns `true` if this `id` is a parent of `objective_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `objective_id` is not found

Raise `NullArgument` – `id` or `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` not found return `false`.

`ObjectiveBank.`**`get_parent_objective_ids`**(*objective_id*)

Gets the parent `Ids` of the given objective.

Parameters **`objective_id`** (`osid.id.Id`) – the `Id` of an objective

Returns the parent `Ids` of the objective

Return type `osid.id.IdList`

Raise `NotFound` – `objective_id` is not found

Raise `NullArgument` – `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_parent_objectives`**(*objective_id*)

Gets the parents of the given objective.

Parameters **`objective_id`** (`osid.id.Id`) – the `Id` of an objective

Returns the parents of the objective

Return type `osid.learning.ObjectiveList`

Raise `NotFound` – `objective_id` is not found

Raise `NullArgument` – `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`is_ancestor_of_objective`**(*id_*, *objective_id*)

Tests if an `Id` is an ancestor of an objective.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`objective_id`** (`osid.id.Id`) – the `Id` of an objective

Returns `true` if this `id` is an ancestor of `objective_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `objective_id` is not found

Raise `NullArgument` – `id` or `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` not found return `false`.

`ObjectiveBank.`**`has_child_objectives`**(*objective_id*)

Tests if an objective has any children.

Parameters **`objective_id`** (`osid.id.Id`) – the `Id` of an objective

Returns `true` if the `objective_id` has children, `false` otherwise

Return type `boolean`

Raise `NotFound` – `objective_id` is not found

Raise `NullArgument` – `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`is_child_of_objective`**(*id_*, *objective_id*)

Tests if an objective is a direct child of another.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`objective_id`** (`osid.id.Id`) – the `Id` of an objective

Returns `true` if the `id` is a child of `objective_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `objective_id` is not found

Raise `NullArgument` – `id` or `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` not found return `false`.

`ObjectiveBank.`**`get_child_objective_ids`**`(`*objective_id*`)`

Gets the child `Ids` of the given objective.

Parameters **`objective_id`** (`osid.id.Id`) – the `Id` to query

Returns the children of the objective

Return type `osid.id.IdList`

Raise `NotFound` – `objective_id` is not found

Raise `NullArgument` – `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_child_objectives`**`(`*objective_id*`)`

Gets the children of the given objective.

Parameters **`objective_id`** (`osid.id.Id`) – the `Id` to query

Returns the children of the objective

Return type `osid.learning.ObjectiveList`

Raise `NotFound` – `objective_id` is not found

Raise `NullArgument` – `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`is_descendant_of_objective`**`(`*id_*, *objective_id*`)`

Tests if an `Id` is a descendant of an objective.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`objective_id`** (`osid.id.Id`) – the `Id` of an objective

Returns `true` if the `id` is a descendant of the `objective_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `objective_id` is not found

Raise `NullArgument` – `id` or `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` is not found return `false`.

`ObjectiveBank.`**`get_objective_node_ids`**(*objective_id*, *ancestor_levels*, *descendant_levels*, *include_siblings*)

Gets a portion of the hierarchy for the given objective.

Parameters

- **`objective_id`** (`osid.id.Id`) – the `Id` to query
- **`ancestor_levels`** (`cardinal`) – the maximum number of ancestor levels to include. A value of 0 returns no parents in the node.
- **`descendant_levels`** (`cardinal`) – the maximum number of descendant levels to include. A value of 0 returns no children in the node.
- **`include_siblings`** (`boolean`) – `true` to include the siblings of the given node, `false` to omit the siblings

Returns a catalog node

Return type `osid.hierarchy.Node`

Raise `NotFound` – `objective_id` not found

Raise `NullArgument` – `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_objective_nodes`**(*objective_id*, *ancestor_levels*, *descendant_levels*, *include_siblings*)

Gets a portion of the hierarchy for the given objective.

Parameters

- **`objective_id`** (`osid.id.Id`) – the `Id` to query
- **`ancestor_levels`** (`cardinal`) – the maximum number of ancestor levels to include. A value of 0 returns no parents in the node.
- **`descendant_levels`** (`cardinal`) – the maximum number of descendant levels to include. A value of 0 returns no children in the node.
- **`include_siblings`** (`boolean`) – `true` to include the siblings of the given node, `false` to omit the siblings

Returns an objective node

Return type `osid.learning.ObjectiveNode`

Raise `NotFound` – `objective_id` not found

Raise `NullArgument` – `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Objective Hierarchy Design Methods

`ObjectiveBank.`**`objective_hierarchy_id`**

Gets the hierarchy `Id` associated with this session.

Returns the hierarchy `Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`objective_hierarchy`**

Gets the hierarchy associated with this session.

Returns the hierarchy associated with this session

Return type `osid.hierarchy.Hierarchy`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`can_modify_objective_hierarchy`**`()`

Tests if this user can change the hierarchy.

A return of true does not guarantee successful authorization. A return of false indicates that it is known performing any update will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer these operations to an unauthorized user.

Returns `false` if changing this hierarchy is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`add_root_objective`**`(`*objective_id*`)`

Adds a root objective.

Parameters **`objective_id`** (`osid.id.Id`) – the `Id` of an objective

Raise `AlreadyExists` – `objective_id` is already in hierarchy

Raise `NotFound` – `objective_id` not found

Raise `NullArgument` – `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`remove_root_objective`**`(`*objective_id*`)`

Removes a root objective.

Parameters **`objective_id`** (`osid.id.Id`) – the `Id` of an objective

Raise `NotFound` – `objective_id` not found

Raise `NullArgument` – `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`add_child_objective`**`(`*objective_id*, *child_id*`)`

Adds a child to an objective.

Parameters

- **objective_id** (`osid.id.Id`) – the `Id` of an objective
- **child_id** (`osid.id.Id`) – the `Id` of the new child

Raise `AlreadyExists` – `objective_id` is already a parent of `child_id`

Raise `NotFound` – `objective_id` or `child_id` not found

Raise `NullArgument` – `objective_id` or `child_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`remove_child_objective`**(*objective_id*, *child_id*)
Removes a child from an objective.

Parameters

- **objective_id** (`osid.id.Id`) – the `Id` of an objective
- **child_id** (`osid.id.Id`) – the `Id` of the new child

Raise `NotFound` – `objective_id` not a parent of `child_id`

Raise `NullArgument` – `objective_id` or `child_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`remove_child_objectives`**(*objective_id*)
Removes all children from an objective.

Parameters **objective_id** (`osid.id.Id`) – the `Id` of an objective

Raise `NotFound` – `objective_id` not found

Raise `NullArgument` – `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Objective Sequencing Methods

`ObjectiveBank.`**`objective_hierarchy_id`**
Gets the hierarchy `Id` associated with this session.

Returns the hierarchy `Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`objective_hierarchy`**
Gets the hierarchy associated with this session.

Returns the hierarchy associated with this session

Return type `osid.hierarchy.Hierarchy`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`can_sequence_objectives`**`()`

Tests if this user can sequence objectives.

A return of true does not guarantee successful authorization. A return of false indicates that it is known performing any update will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer these operations to an unauthorized user.

Returns `false` if sequencing objectives is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`move_objective_ahead`**`(`*parent_objective_id*, *reference_objective_id*, *objective_id*`)`

Moves an objective ahead of a refrence objective under the given parent.

Parameters

- **`parent_objective_id`** (`osid.id.Id`) – the `Id` of the parent objective
- **`reference_objective_id`** (`osid.id.Id`) – the `Id` of the objective
- **`objective_id`** (`osid.id.Id`) – the `Id` of the objective to move ahead of `reference_objective_id`

Raise `NotFound` – `parent_objective_id, reference_objective_id,` or `objective_id` not found, or `reference_objective_id` or `objective_id` is not a child of `parent_objective_id`

Raise `NullArgument` – `parent_objective_id, reference_objective_id,` or `id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`move_objective_behind`**`(`*parent_objective_id*, *reference_objective_id*, *objective_id*`)`

Moves an objective behind a refrence objective under the given parent.

Parameters

- **`parent_objective_id`** (`osid.id.Id`) – the `Id` of the parent objective
- **`reference_objective_id`** (`osid.id.Id`) – the `Id` of the objective
- **`objective_id`** (`osid.id.Id`) – the `Id` of the objective to move behind `reference_objective_id`

Raise `NotFound` – `parent_objective_id, reference_objective_id,` or `objective_id` not found, or `reference_objective_id` or `objective_id` is not a child of `parent_objective_id`

Raise `NullArgument` – `parent_objective_id, reference_objective_id,` or `id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`sequence_objectives`**(*parent_objective_id*, *objective_ids*)
Sequences a set of objectives under a parent.

Parameters

- **`parent_objective_id`** (`osid.id.Id`) – the `Id` of the parent objective
- **`objective_ids`** (`osid.id.Id[]`) – the `Id` of the objectives

Raise `NotFound` – `parent_id` or an `objective_id` not found, or an `objective_id` is not a child of `parent_objective_id`

Raise `NullArgument` – `paren_objectivet_id` or `objective_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Objective Objective Bank Methods

`ObjectiveBank.`**`can_lookup_objective_objective_bank_mappings`**()
Tests if this user can perform lookups of objective/objective bank mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known lookup methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if looking up mappings is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`use_comparative_objective_bank_view`**()
The returns from the lookup methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`use_plenary_objective_bank_view`**()
A complete view of the `Activity` and `ObjectiveBank` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`get_objective_ids_by_objective_bank`**(*objective_bank_id*)
Gets the list of `Objective Ids` associated with an `ObjectiveBank`.

Parameters **`objective_bank_id`** (`osid.id.Id`) – `Id` of the `ObjectiveBank`

Returns list of related objectives

Return type `osid.id.IdList`

Raise `NotFound` – `objective_bank_id` is not found

Raise `NullArgument` – `objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_objectives_by_objective_bank`**(*objective_bank_id*)

Gets the list of `Objectives` associated with an `ObjectiveBank`.

Parameters **`objective_bank_id`** (`osid.id.Id`) – `Id` of the `ObjectiveBank`

Returns list of related objective `Ids`

Return type `osid.learning.ObjectiveList`

Raise `NotFound` – `objective_bank_id` is not found

Raise `NullArgument` – `objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_objective_ids_by_objective_banks`**(*objective_bank_ids*)

Gets the list of `Objective Ids` corresponding to a list of `ObjectiveBanks`.

Parameters **`objective_bank_ids`** (`osid.id.IdList`) – list of objective bank `Ids`

Returns list of objective `Ids`

Return type `osid.id.IdList`

Raise `NullArgument` – `objective_bank_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_objectives_by_objective_banks`**(*objective_bank_ids*)

Gets the list of `Objectives` corresponding to a list of `ObjectiveBanks`.

Parameters **`objective_bank_ids`** (`osid.id.IdList`) – list of objective bank `Ids`

Returns list of objectives

Return type `osid.learning.ObjectiveList`

Raise `NullArgument` – `objective_bank_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_objective_bank_ids_by_objective`**(*objective_id*)

Gets the list of `ObjectiveBank Ids` mapped to an `Objective`.

Parameters **`objective_id`** (`osid.id.Id`) – `Id` of an `Objective`

Returns list of objective bank `Ids`

Return type `osid.id.IdList`

Raise `NotFound` – `objective_id` is not found

Raise `NullArgument` – `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_objective_banks_by_objective`**`(`*objective_id*`)`

Gets the list of `ObjectiveBanks` mapped to an `Objective`.

Parameters **`objective_id`** (`osid.id.Id`) – `Id` of an `Objective`

Returns list of objective banks

Return type `osid.learning.ObjectiveBankList`

Raise `NotFound` – `objective_id` is not found

Raise `NullArgument` – `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Objective Objective Bank Assignment Methods

`ObjectiveBank.`**`can_assign_objectives`**`()`

Tests if this user can alter objective/objective bank mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known mapping methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer assignment operations to unauthorized users.

Returns `false` if mapping is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`can_assign_objectives_to_objective_bank`**`(`*objective_bank_id*`)`

Tests if this user can alter objective/objective bank mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known mapping methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer assignment operations to unauthorized users.

Parameters **`objective_bank_id`** (`osid.id.Id`) – the `Id` of the `ObjectiveBank`

Returns `false` if mapping is not authorized, `true` otherwise

Return type `boolean`

Raise `NullArgument` – `objective_bank_id` is `null`

compliance: mandatory – This method must be implemented.

ObjectiveBank.**get_assignable_objective_bank_ids**(*objective_bank_id*)

Gets a list of objective banks including and under the given objective bank node in which any activity can be assigned.

Parameters **objective_bank_id** (osid.id.Id) – the Id of the ObjectiveBank

Returns list of assignable objective bank Ids

Return type osid.id.IdList

Raise NullArgument – objective_bank_id is null

Raise OperationFailed – unable to complete request

compliance: mandatory – This method must be implemented.

ObjectiveBank.**get_assignable_objective_bank_ids_for_objective**(*objective_bank_id*, *objective_id*)

Gets a list of objective banks including and under the given objective bank node in which a specific objective can be assigned.

Parameters

- **objective_bank_id** (osid.id.Id) – the Id of the ObjectiveBank
- **objective_id** (osid.id.Id) – the Id of the Objective

Returns list of assignable objective bank Ids

Return type osid.id.IdList

Raise NullArgument – objective_id or objective_bank_id is null

Raise OperationFailed – unable to complete request

compliance: mandatory – This method must be implemented.

ObjectiveBank.**assign_objective_to_objective_bank**(*objective_id*, *objective_bank_id*)

Adds an existing Objective to an ObjectiveBank.

Parameters

- **objective_id** (osid.id.Id) – the Id of the Objective
- **objective_bank_id** (osid.id.Id) – the Id of the ObjectiveBank

Raise AlreadyExists – objective_id already mapped to objective_bank_id

Raise NotFound – objective_id or objective_bank_id not found

Raise NullArgument – objective_id or objective_bank_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

ObjectiveBank.**unassign_objective_from_objective_bank**(*objective_id*, *objective_bank_id*)

Removes an Objective from an ObjectiveBank.

Parameters

- **objective_id** (osid.id.Id) – the Id of the Objective

- **objective_bank_id** (osid.id.Id) – the Id of the ObjectiveBank

Raise NotFound – objective_id or objective_bank_id not found or objective_id not mapped to objective_bank_id

Raise NullArgument – objective_id or objective_bank_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

ObjectiveBank.**reassign_proficiency_to_objective_bank**(*objective_id*, *from_objective_bank_id*, *to_objective_bank_id*)

Moves an Objective from one ObjectiveBank to another.

Mappings to other ObjectiveBanks are unaffected.

Parameters

- **objective_id** (osid.id.Id) – the Id of the Objective
- **from_objective_bank_id** (osid.id.Id) – the Id of the current ObjectiveBank
- **to_objective_bank_id** (osid.id.Id) – the Id of the destination ObjectiveBank

Raise NotFound – objective_id, from_objective_bank_id, or to_objective_bank_id not found or objective_id not mapped to from_objective_bank_id

Raise NullArgument – objective_id, from_objective_bank_id, or to_objective_bank_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Objective Requisite Methods

ObjectiveBank.**objective_bank_id**

Gets the ObjectiveBank Id associated with this session.

Returns the ObjectiveBank Id associated with this session

Return type osid.id.Id

compliance: mandatory – This method must be implemented.

ObjectiveBank.**objective_bank**

Gets the ObjectiveBank associated with this session.

Returns the obective bank

Return type osid.learning.ObjectiveBank

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`can_lookup_objective_prerequisites`**`()`

Tests if this user can perform `Objective` lookups.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if lookup methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`use_comparative_objective_view`**`()`

The returns from the lookup methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`use_plenary_objective_view`**`()`

A complete view of the `Objective` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`use_federated_objective_bank_view`**`()`

Federates the view for methods in this session.

A federated view will include proficiencies in objective banks which are children of this objective bank in the obective bank hierarchy.

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`use_isolated_objective_bank_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts lookups to this objective bank only.

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`get_requisite_objectives`**`(`*objective_id*`)`

Gets a list of `Objectives` that are the immediate requisites for the given `Objective`.

In plenary mode, the returned list contains all of the immediate requisites, or an error results if an `Objective` is not found or inaccessible. Otherwise, inaccessible `Objectives` may be omitted from the list and may present the elements in any order including returning a unique set.

Parameters **`objective_id`** (`osid.id.Id`) – `Id` of the `Objective`

Returns the returned requisite `Objectives`

Return type `osid.learning.ObjectiveList`

Raise `NotFound` – `objective_id` not found

Raise `NullArgument` – `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`get_all_requisite_objectives`**(*objective_id*)

Gets a list of `Objectives` that are the requisites for the given `Objective` including the requistes of the requisites, and so on.

In plenary mode, the returned list contains all of the immediate requisites, or an error results if an `Objective` is not found or inaccessible. Otherwise, inaccessible `Objectives` may be omitted from the list and may present the elements in any order including returning a unique set.

Parameters **`objective_id`** (`osid.id.Id`) – `Id` of the `Objective`

Returns the returned `Objective` list

Return type `osid.learning.ObjectiveList`

Raise `NotFound` – `objective_id` not found

Raise `NullArgument` – `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_dependent_objectives`**(*objective_id*)

Gets a list of `Objectives` that require the given `Objective`.

In plenary mode, the returned list contains all of the immediate requisites, or an error results if an Objective is not found or inaccessible. Otherwise, inaccessible `Objectives` may be omitted from the list and may present the elements in any order including returning a unique set.

Parameters **`objective_id`** (`osid.id.Id`) – `Id` of the `Objective`

Returns the returned `Objective` list

Return type `osid.learning.ObjectiveList`

Raise `NotFound` – `objective_id` not found

Raise `NullArgument` – `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`is_objective_required`**(*objective_id*, *required_objective_id*)

Tests if an objective is required before proceeding with an objective.

One objective may indirectly depend on another objective by way of one or more other objectives.

Parameters

- **`objective_id`** (`osid.id.Id`) – `Id` of the dependent `Objective`
- **`required_objective_id`** (`osid.id.Id`) – `Id` of the required `Objective`

Returns `true` if `objective_id` depends on `required_objective_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `objective_id` not found

Raise `NullArgument` – `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_equivalent_objectives`**`(`*objective_id*`)`

Gets a list of `Objectives` that are equivalent to the given `Objective` for the purpose of requisites.

An equivalent objective can satisfy the given objective. In plenary mode, the returned list contains all of the equivalent requisites, or an error results if an Objective is not found or inaccessible. Otherwise, inaccessible `Objectives` may be omitted from the list and may present the elements in any order including returning a unique set.

Parameters **`objective_id`** (`osid.id.Id`) – `Id` of the `Objective`

Returns the returned `Objective` list

Return type `osid.learning.ObjectiveList`

Raise `NotFound` – `objective_id` not found

Raise `NullArgument` – `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Objective Requisite Assignment Methods

`ObjectiveBank.`**`objective_bank_id`**

Gets the `ObjectiveBank Id` associated with this session.

Returns the `ObjectiveBank Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`objective_bank`**

Gets the `ObjectiveBank` associated with this session.

Returns the obective bank

Return type `osid.learning.ObjectiveBank`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`can_assign_requisites`**`()`

Tests if this user can manage objective requisites.

A return of true does not guarantee successful authorization. A return of false indicates that it is known mapping methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer assignment operations to unauthorized users.

Returns `false` if mapping is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`assign_objective_requisite`**(*objective_id*, *requisite_objective_id*)

Creates a requirement dependency between two `Objectives`.

Parameters

- **`objective_id`** (`osid.id.Id`) – the `Id` of the dependent `Objective`
- **`requisite_objective_id`** (`osid.id.Id`) – the `Id` of the required `Objective`

Raise `AlreadyExists` – `objective_id` already mapped to `requisite_objective_id`

Raise `NotFound` – `objective_id` or `requisite_objective_id` not found

Raise `NullArgument` – `objective_id` or `requisite_objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`unassign_objective_requisite`**(*objective_id*, *requisite_objective_id*)

Removes an `Objective` requisite from an `Objective`.

Parameters

- **`objective_id`** (`osid.id.Id`) – the `Id` of the `Objective`
- **`requisite_objective_id`** (`osid.id.Id`) – the `Id` of the required `Objective`

Raise `NotFound` – `objective_id` or `requisite_objective_id` not found or `objective_id` not mapped to `requisite_objective_id`

Raise `NullArgument` – `objective_id` or `requisite_objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`assign_equivalent_objective`**(*objective_id*, *equivalent_objective_id*)

Makes an objective equivalent to another objective for the purposes of satisfying a requisite.

Parameters

- **`objective_id`** (`osid.id.Id`) – the `Id` of the principal `Objective`
- **`equivalent_objective_id`** (`osid.id.Id`) – the `Id` of the equivalent `Objective`

Raise `AlreadyExists` – `objective_id` already mapped to `equiavelnt_objective_id`

Raise `NotFound` – `objective_id` or `equivalent_objective_id` not found

Raise `NullArgument` – `objective_id` or `equivalent_objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`unassign_equivalent_objective`**`(`*objective_id, equivalent_objective_id*`)`

Removes an `Objective` requisite from an `Objective`.

Parameters

- **`objective_id`** (`osid.id.Id`) – the `Id` of the principal `Objective`
- **`equivalent_objective_id`** (`osid.id.Id`) – the `Id` of the equivalent `Objective`

Raise `NotFound` – `objective_id` or `equivalent_objective_id` not found or `objective_id` is already equivalent to `equivalent_objective_id`

Raise `NullArgument` – `objective_id` or `equivalent_objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Activity Lookup Methods

`ObjectiveBank.`**`objective_bank_id`**

Gets the `ObjectiveBank Id` associated with this session.

Returns the `ObjectiveBank Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`objective_bank`**

Gets the `ObjectiveBank` associated with this session.

Returns the obective bank

Return type `osid.learning.ObjectiveBank`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`can_lookup_activities`**`()`

Tests if this user can perform `Activity` lookups.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if lookup methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`use_comparative_activity_view`**`()`

The returns from the lookup methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`use_plenary_activity_view`**`()`

A complete view of the `Activity` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`use_federated_objective_bank_view`**`()`

Federates the view for methods in this session.

A federated view will include proficiencies in objective banks which are children of this objective bank in the obective bank hierarchy.

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`use_isolated_objective_bank_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts lookups to this objective bank only.

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`get_activity`**`(`*activity_id*`)`

Gets the `Activity` specified by its `Id`.

In plenary mode, the exact `Id` is found or a `NotFound` results. Otherwise, the returned `Activity` may have a different `Id` than requested, such as the case where a duplicate `Id` was assigned to a `Activity` and retained for compatibility.

Parameters **`activity_id`** (`osid.id.Id`) – `Id` of the `Activity`

Returns the activity

Return type `osid.learning.Activity`

Raise `NotFound` – `activity_id` not found

Raise `NullArgument` – `activity_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`get_activities_by_ids`**`(`*activity_ids*`)`

Gets an `ActivityList` corresponding to the given `IdList`.

In plenary mode, the returned list contains all of the activities specified in the `Id` list, in the order of the list, including duplicates, or an error results if an `Id` in the supplied list is not found or inaccessible. Otherwise, inaccessible `Activities` may be omitted from the list and may present the elements in any order including returning a unique set.

Parameters **`activity_ids`** (`osid.id.IdList`) – the list of `Ids` to retrieve

Returns the returned `Activity` list

Return type `osid.learning.ActivityList`

Raise `NotFound` – an `Id was` not found

Raise `NullArgument` – `activity_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_activities_by_genus_type`**`(`*activity_genus_type*`)`

Gets an `ActivityList` corresponding to the given activity genus `Type` which does not include activities of genus types derived from the specified `Type`.

In plenary mode, the returned list contains all known activities or an error results. Otherwise, the returned list may contain only those activities that are accessible through this session.

Parameters **`activity_genus_type`** (`osid.type.Type`) – an activity genus type

Returns the returned `Activity` list

Return type `osid.learning.ActivityList`

Raise `NullArgument` – `activity_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_activities_by_parent_genus_type`**`(`*activity_genus_type*`)`

Gets an `ActivityList` corresponding to the given activity genus `Type` and include any additional activity with genus types derived from the specified `Type`.

In plenary mode, the returned list contains all known activities or an error results. Otherwise, the returned list may contain only those activities that are accessible through this session.

Parameters **`activity_genus_type`** (`osid.type.Type`) – an activity genus type

Returns the returned `Activity` list

Return type `osid.learning.ActivityList`

Raise `NullArgument` – `activity_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_activities_by_record_type`**`(`*activity_record_type*`)`

Gets a `ActivityList` containing the given activity record `Type`.

In plenary mode, the returned list contains all known activities or an error results. Otherwise, the returned list may contain only those activities that are accessible through this session.

Parameters **`activity_record_type`** (`osid.type.Type`) – an activity record type

Returns the returned `Activity` list

Return type `osid.learning.ActivityList`

Raise `NullArgument` – `activity_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_activities_for_objective`**(*objective_id*)

Gets the activities for the given objective.

In plenary mode, the returned list contains all of the activities mapped to the objective `Id` or an error results if an Id in the supplied list is not found or inaccessible. Otherwise, inaccessible `Activities` may be omitted from the list and may present the elements in any order including returning a unique set.

Parameters **`objective_id`** (`osid.id.Id`) – `Id` of the `Objective`

Returns list of enrollments

Return type `osid.learning.ActivityList`

Raise `NotFound` – `objective_id` not found

Raise `NullArgument` – `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`get_activities_for_objectives`**(*objective_ids*)

Gets the activities for the given objectives.

In plenary mode, the returned list contains all of the activities specified in the objective `Id` list, in the order of the list, including duplicates, or an error results if a course offering `Id` in the supplied list is not found or inaccessible. Otherwise, inaccessible `Activities` may be omitted from the list and may present the elements in any order including returning a unique set.

Parameters **`objective_ids`** (`osid.id.IdList`) – list of objective `Ids`

Returns list of activities

Return type `osid.learning.ActivityList`

Raise `NotFound` – an `objective_id` not found

Raise `NullArgument` – `objective_id_list` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`get_activities_by_asset`**(*asset_id*)

Gets the activities for the given asset.

In plenary mode, the returned list contains all of the activities mapped to the asset `Id` or an error results if an `Id` in the supplied list is not found or inaccessible. Otherwise, inaccessible `Activities` may be omitted from the list and may present the elements in any order including returning a unique set.

Parameters **`asset_id`** (`osid.id.Id`) – `Id` of an `Asset`

Returns list of activities

Return type `osid.learning.ActivityList`

Raise `NotFound` – `asset_id` not found

Raise `NullArgument` – `asset_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`get_activities_by_assets`**(*asset_ids*)

Gets the activities for the given asset.

In plenary mode, the returned list contains all of the activities mapped to the asset `Id` or an error results if an `Id` in the supplied list is not found or inaccessible. Otherwise, inaccessible `Activities` may be omitted from the list and may present the elements in any order including returning a unique set.

Parameters **`asset_ids`** (`osid.id.IdList`) – `Ids` of `Assets`

Returns list of activities

Return type `osid.learning.ActivityList`

Raise `NotFound` – an `asset_id` not found

Raise `NullArgument` – `asset_id_list` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`activities`**

Gets all `Activities`.

In plenary mode, the returned list contains all known activites or an error results. Otherwise, the returned list may contain only those activities that are accessible through this session.

Returns a `ActivityList`

Return type `osid.learning.ActivityList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Activity Admin Methods

`ObjectiveBank.`**`objective_bank_id`**

Gets the `ObjectiveBank Id` associated with this session.

Returns the `ObjectiveBank Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`objective_bank`**

Gets the `ObjectiveBank` associated with this session.

Returns the obective bank

Return type `osid.learning.ObjectiveBank`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

ObjectiveBank.**can_create_activities**()

Tests if this user can create Activities.

A return of true does not guarantee successful authorization. A return of false indicates that it is known creating an Activity will result in a PermissionDenied. This is intended as a hint to an application that may opt not to offer create operations to an unauthorized user.

Returns false if Activity creation is not authorized, true otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

ObjectiveBank.**can_create_activity_with_record_types**(*activity_record_types*)

Tests if this user can create a single Activity using the desired record types.

While LearningManager.getActivityRecordTypes() can be used to examine which records are supported, this method tests which record(s) are required for creating a specific Activity. Providing an empty array tests if an Activity can be created with no records.

Parameters **activity_record_types** (osid.type.Type[]) – array of activity record types

Returns true if Activity creation using the specified record Types is supported, false otherwise

Return type boolean

Raise NullArgument – activity_record_types is null

compliance: mandatory – This method must be implemented.

ObjectiveBank.**get_activity_form_for_create**(*objective_id*, *activity_record_types*)

Gets the activity form for creating new activities.

A new form should be requested for each create transaction.

Parameters

- **objective_id** (osid.id.Id) – the Id of the Objective
- **activity_record_types** (osid.type.Type[]) – array of activity record types

Returns the activity form

Return type osid.learning.ActivityForm

Raise NotFound – objective_id is not found

Raise NullArgument – objective_id or activity_record_types is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

Raise Unsupported – unable to get form for requested record types

compliance: mandatory – This method must be implemented.

ObjectiveBank.**create_activity**(*activity_form*)

Creates a new Activity.

Parameters **activity_form** (osid.learning.ActivityForm) – the form for this Activity

Returns the new Activity

Return type `osid.learning.Activity`

Raise `IllegalState` – `activity_form` already used in a create transaction

Raise `InvalidArgument` – one or more of the form elements is invalid

Raise `NullArgument` – `activity_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `activity_form` did not originate from `get_activity_form_for_create()`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`can_update_activities`**`()`

Tests if this user can update `Activities`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating an `Activity` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer update operations to an unauthorized user.

Returns `false` if activity modification is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_activity_form_for_update`**`(`*activity_id*`)`

Gets the activity form for updating an existing activity.

A new activity form should be requested for each update transaction.

Parameters **`activity_id`** (`osid.id.Id`) – the `Id` of the `Activity`

Returns the activity form

Return type `osid.learning.ActivityForm`

Raise `NotFound` – `activity_id` is not found

Raise `NullArgument` – `activity_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`update_activity`**`(`*activity_form*`)`

Updates an existing activity,.

Parameters **`activity_form`** (`osid.learning.ActivityForm`) – the form containing the elements to be updated

Raise `IllegalState` – `activity_form` already used in an update transaction

Raise `InvalidArgument` – the form contains an invalid value

Raise `NullArgument` – `activity_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `activity_form` did not originate from `get_activity_form_for_update()`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`can_delete_activities`**`()`

Tests if this user can delete `Activities`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting an `Activity` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer delete operations to an unauthorized user.

Returns `false` if `Activity` deletion is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`delete_activity`**`(`*activity_id*`)`

Deletes the `Activity` identified by the given `Id`.

Parameters **`activity_id`** (`osid.id.Id`) – the `Id` of the `Activity` to delete

Raise `NotFound` – an `Activity` was not found identified by the given `Id`

Raise `NullArgument` – `activity_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`can_manage_activity_aliases`**`()`

Tests if this user can manage `Id` aliases for activities.

A return of true does not guarantee successful authorization. A return of false indicates that it is known changing an alias will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer alias operations to an unauthorized user.

Returns `false` if `Activity` aliasing is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`alias_activity`**`(`*activity_id*, *alias_id*`)`

Adds an `Id` to an `Activity` for the purpose of creating compatibility.

The primary `Id` of the `Activity` is determined by the provider. The new `Id` performs as an alias to the primary `Id`. If the alias is a pointer to another activity, it is reassigned to the given activity `Id`.

Parameters

- **`activity_id`** (`osid.id.Id`) – the `Id` of an `Activity`
- **`alias_id`** (`osid.id.Id`) – the alias `Id`

Raise `AlreadyExists` – `alias_id` is already assigned

Raise `NotFound` – `activity_id` not found

Raise `NullArgument` – `activity_id` or `alias_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Activity Objective Bank Methods

`ObjectiveBank.`**`can_lookup_activity_objective_bank_mappings`**`()`

Tests if this user can perform lookups of activity/objective bank mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known lookup methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if looking up mappings is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`use_comparative_objective_bank_view`**`()`

The returns from the lookup methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`use_plenary_objective_bank_view`**`()`

A complete view of the `Activity` and `ObjectiveBank` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`get_activity_ids_by_objective_bank`**`(`*objective_bank_id*`)`

Gets the list of `Activity Ids` associated with an `ObjectiveBank`.

Parameters **`objective_bank_id`** (`osid.id.Id`) – `Id` of the `ObjectiveBank`

Returns list of related activity `Ids`

Return type `osid.id.IdList`

Raise `NotFound` – `objective_bank_id` is not found

Raise `NullArgument` – `objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_activities_by_objective_bank`**`(`*objective_bank_id*`)`

Gets the list of `Activities` associated with an `ObjectiveBank`.

Parameters **`objective_bank_id`** (`osid.id.Id`) – `Id` of the `ObjectiveBank`

Returns list of related activities

Return type `osid.learning.ActivityList`

Raise `NotFound` – `objective_bank_id` is not found

Raise `NullArgument` – `objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_activity_ids_by_objective_banks`**(*objective_bank_ids*)

Gets the list of `Activity Ids` corresponding to a list of `ObjectiveBanks`.

Parameters **`objective_bank_ids`** (`osid.id.IdList`) – list of objective bank `Ids`

Returns list of activity `Ids`

Return type `osid.id.IdList`

Raise `NullArgument` – `objective_bank_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_activities_by_objective_banks`**(*objective_bank_ids*)

Gets the list of `Activities` corresponding to a list of `ObjectiveBanks`.

Parameters **`objective_bank_ids`** (`osid.id.IdList`) – list of objective bank `Ids`

Returns list of activities

Return type `osid.learning.ActivityList`

Raise `NullArgument` – `objective_bank_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_objective_bank_ids_by_activity`**(*activity_id*)

Gets the list of `ObjectiveBank Ids` mapped to a `Activity`.

Parameters **`activity_id`** (`osid.id.Id`) – `Id` of a `Activity`

Returns list of objective bank `Ids`

Return type `osid.id.IdList`

Raise `NotFound` – `activity_id` is not found

Raise `NullArgument` – `activity_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_objective_banks_by_activity`**(*activity_id*)

Gets the list of `ObjectiveBanks` mapped to a `Activity`.

Parameters **`activity_id`** (`osid.id.Id`) – `Id` of a `Activity`

Returns list of objective bank `Ids`

Return type `osid.learning.ObjectiveBankList`

Raise `NotFound` – `activity_id` is not found

Raise `NullArgument` – `activity_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Activity Objective Bank Assignment Methods

`ObjectiveBank.`**`can_assign_activities`**`()`

Tests if this user can alter activity/objective bank mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known mapping methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer assignment operations to unauthorized users.

Returns `false` if mapping is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`can_assign_activities_to_objective_bank`**`(`*objective_bank_id*`)`

Tests if this user can alter activity/objective bank mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known mapping methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer assignment operations to unauthorized users.

Parameters **`objective_bank_id`** (`osid.id.Id`) – the `Id` of the `ObjectiveBank`

Returns `false` if mapping is not authorized, `true` otherwise

Return type `boolean`

Raise `NullArgument` – `objective_bank_id` is `null`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_assignable_objective_bank_ids`**`(`*objective_bank_id*`)`

Gets a list of objective banks including and under the given objective bank node in which any activity can be assigned.

Parameters **`objective_bank_id`** (`osid.id.Id`) – the `Id` of the `ObjectiveBank`

Returns list of assignable objective bank `Ids`

Return type `osid.id.IdList`

Raise `NullArgument` – `objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_assignable_objective_bank_ids_for_activity`**`(`*objective_bank_id, activity_id*`)`

Gets a list of objective banks including and under the given objective bank node in which a specific activity can be assigned.

Parameters

- **`objective_bank_id`** (`osid.id.Id`) – the `Id` of the `ObjectiveBank`
- **`activity_id`** (`osid.id.Id`) – the `Id` of the `Activity`

Returns list of assignable objective bank `Ids`

Return type `osid.id.IdList`

Raise `NullArgument` – `activity_id` or `objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`assign_activity_to_objective_bank`**(*activity_id*, *objective_bank_id*)

Adds an existing `Activity` to a `ObjectiveBank`.

Parameters

- **`activity_id`** (`osid.id.Id`) – the `Id` of the `Activity`
- **`objective_bank_id`** (`osid.id.Id`) – the `Id` of the `ObjectiveBank`

Raise `AlreadyExists` – `activity_id` already mapped to `objective_bank_id`

Raise `NotFound` – `activity_id` or `objective_bank_id` not found

Raise `NullArgument` – `activity_id` or `objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`unassign_activity_from_objective_bank`**(*activity_id*, *objective_bank_id*)

Removes a `Activity` from a `ObjectiveBank`.

Parameters

- **`activity_id`** (`osid.id.Id`) – the `Id` of the `Activity`
- **`objective_bank_id`** (`osid.id.Id`) – the `Id` of the `ObjectiveBank`

Raise `NotFound` – `activity_id` or `objective_bank_id` not found or `activity_id` not mapped to `objective_bank_id`

Raise `NullArgument` – `activity_id` or `objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`reassign_activity_to_objective_bank`**(*activity_id*, *from_objective_bank_id*, *to_objective_bank_id*)

Moves an `Activity` from one `ObjectiveBank` to another.

Mappings to other `ObjectiveBanks` are unaffected.

Parameters

- **`activity_id`** (`osid.id.Id`) – the `Id` of the `Activity`
- **`from_objective_bank_id`** (`osid.id.Id`) – the `Id` of the current `ObjectiveBank`
- **`to_objective_bank_id`** (`osid.id.Id`) – the `Id` of the destination `ObjectiveBank`

Raise `NotFound` – `activity_id`, `from_objective_bank_id`, or `to_objective_bank_id` not found or `activity_id` not mapped to `from_objective_bank_id`

Raise `NullArgument` – `activity_id`, `from_objective_bank_id`, or `to_objective_bank_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Proficiency Lookup Methods

`ObjectiveBank.`**`objective_bank_id`**

Gets the `ObjectiveBank Id` associated with this session.

Returns the `ObjectiveBank Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`objective_bank`**

Gets the `ObjectiveBank` associated with this session.

Returns the obective bank

Return type `osid.learning.ObjectiveBank`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`can_lookup_proficiencies`**`()`

Tests if this user can perform `Proficiency` lookups.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may not offer lookup operations to unauthorized users.

Returns `false` if lookup methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`use_comparative_proficiency_view`**`()`

The returns from the lookup methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`use_plenary_proficiency_view`**`()`

A complete view of the `Proficiency` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

ObjectiveBank.**use_federated_objective_bank_view**()

Federates the view for methods in this session.

A federated view will include proficiencies in objective banks which are children of this objective bank in the obective bank hierarchy.

compliance: mandatory – This method is must be implemented.

ObjectiveBank.**use_isolated_objective_bank_view**()

Isolates the view for methods in this session.

An isolated view restricts lookups to this objective bank only.

compliance: mandatory – This method is must be implemented.

ObjectiveBank.**use_effective_proficiency_view**()

Only proficiencies whose effective dates are current are returned by methods in this session.

compliance: mandatory – This method is must be implemented.

ObjectiveBank.**use_any_effective_proficiency_view**()

All proficienies of any effective dates are returned by methods in this session.

compliance: mandatory – This method is must be implemented.

ObjectiveBank.**get_proficiency**(*proficiency_id*)

Gets the `Proficiency` specified by its `Id`.

Parameters proficiency_id (`osid.id.Id`) – the `Id` of the `Proficiency` to retrieve

Returns the returned `Proficiency`

Return type `osid.learning.Proficiency`

Raise `NotFound` – no `Proficiency` found with the given `Id`

Raise `NullArgument` – `proficiency_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

ObjectiveBank.**get_proficiencies_by_ids**(*proficiency_ids*)

Gets a `ProficiencyList` corresponding to the given `IdList`.

Parameters proficiency_ids (`osid.id.IdList`) – the list of `Ids` to retrieve

Returns the returned `Proficiency` list

Return type `osid.learning.ProficiencyList`

Raise `NotFound` – an `Id` was not found

Raise `NullArgument` – `proficiency_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

ObjectiveBank.**get_proficiencies_by_genus_type**(*proficiency_genus_type*)

Gets a `ProficiencyList` corresponding to the given proficiency genus `Type` which does not include proficiencies of types derived from the specified `Type`.

Parameters **proficiency_genus_type** (`osid.type.Type`) – a proficiency genus type

Returns the returned `Proficiency` list

Return type `osid.learning.ProficiencyList`

Raise `NullArgument` – `proficiency_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**get_proficiencies_by_parent_genus_type**(*proficiency_genus_type*)

Gets a `ProficiencyList` corresponding to the given proficiency genus `Type` and include any additional proficiencies with genus types derived from the specified `Type`.

Parameters **proficiency_genus_type** (`osid.type.Type`) – a proficiency genus type

Returns the returned `Proficiency` list

Return type `osid.learning.ProficiencyList`

Raise `NullArgument` – `proficiency_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**get_proficiencies_by_record_type**(*proficiency_record_type*)

Gets a `ProficiencyList` containing the given proficiency record `Type`.

Parameters **proficiency_record_type** (`osid.type.Type`) – a proficiency record type

Returns the returned `Proficiency` list

Return type `osid.learning.ProficiencyList`

Raise `NullArgument` – `proficiency_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**get_proficiencies_on_date**(*from_*, *to*)

Gets a `ProficiencyList` effecyive during the entire given date range inclusive but not confined to the date range.

Parameters

- **from** (`osid.calendaring.DateTime`) – starting date
- **to** (`osid.calendaring.DateTime`) – ending date

Returns the returned `Proficiency` list

Return type `osid.learning.ProficiencyList`

Raise `InvalidArgument` – `from` is greater than `to`

Raise `NullArgument` – `from` or `to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_proficiencies_by_genus_type_on_date`**(*proficiency_genus_type*, *from_*, *to*)

Gets a `ProficiencyList` of the given proficiency genus type effective during the entire given date range inclusive but not confined to the date range.

Parameters

- **`proficiency_genus_type`** (`osid.type.Type`) – a proficiency genus type
- **`from`** (`osid.calendaring.DateTime`) – starting date
- **`to`** (`osid.calendaring.DateTime`) – ending date

Returns the returned `Proficiency` list

Return type `osid.learning.ProficiencyList`

Raise `InvalidArgument` – `from` is greater than `to`

Raise `NullArgument` – `proficiency_genus_type, from`, or `to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_proficiencies_for_objective`**(*objective_id*)

Gets a `ProficiencyList` relating to the given objective.

Parameters **`objective_id`** (`osid.id.Id`) – an objective `Id`

Returns the returned `Proficiency` list

Return type `osid.learning.ProficiencyList`

Raise `NullArgument` – `objective_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_proficiencies_for_objective_on_date`**(*objective_id*, *from_*, *to*)

Gets a `ProficiencyList` relating to the given objective effective during the entire given date range inclusive but not confined to the date range.

Parameters

- **`objective_id`** (`osid.id.Id`) – an objective `Id`
- **`from`** (`osid.calendaring.DateTime`) – starting date
- **`to`** (`osid.calendaring.DateTime`) – ending date

Returns the returned `Proficiency` list

Return type `osid.learning.ProficiencyList`

Raise `InvalidArgument` – `from` is greater than `to`

Raise `NullArgument` – `objective_id, from` or `to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_proficiencies_by_genus_type_for_objective`**(*objective_id*, *proficiency_genus_type*)

Gets a `ProficiencyList` relating to the given objective and proficiency genus `Type`.

Parameters

- **`objective_id`** (`osid.id.Id`) – an objective `Id`
- **`proficiency_genus_type`** (`osid.type.Type`) – a proficiency genus type

Returns the returned `Proficiency` list

Return type `osid.learning.ProficiencyList`

Raise `NullArgument` – `objective_id` or `proficiency_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_proficiencies_by_genus_type_for_objective_on_date`**(*objective_id*, *proficiency_genus_type*, *from_*, *to*)

Gets a `ProficiencyList` of the given proficiency genus type relating to the given objective effective during the entire given date range inclusive but not confined to the date range.

Parameters

- **`objective_id`** (`osid.id.Id`) – an objective `Id`
- **`proficiency_genus_type`** (`osid.type.Type`) – a proficiency genus type
- **`from`** (`osid.calendaring.DateTime`) – starting date
- **`to`** (`osid.calendaring.DateTime`) – ending date

Returns the returned `Proficiency` list

Return type `osid.learning.ProficiencyList`

Raise `InvalidArgument` – `from` is greater than `to`

Raise `NullArgument` – `objective_id, proficiency_genus_type, from,` or `to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_proficiencies_for_objectives`**(*objective_ids*)

Gets a `ProficiencyList` relating to the given objectives.

Parameters **`objective_ids`** (`osid.id.IdList`) – the objective `Ids`

Returns the returned `Proficiency` list

Return type `osid.learning.ProficiencyList`

Raise `NullArgument` – `objective_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_proficiencies_for_resource`**(*resource_id*)

Gets a `ProficiencyList` relating to the given resource.

Parameters **`resource_id`** (`osid.id.Id`) – a resource `Id`

Returns the returned `Proficiency` list

Return type `osid.learning.ProficiencyList`

Raise `NullArgument` – `resource_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_proficiencies_for_resource_on_date`**(*resource_id*, *from_*, *to*)

Gets a `ProficiencyList` relating to the given resource effective during the entire given date range inclusive but not confined to the date range.

Parameters

- **`resource_id`** (`osid.id.Id`) – a resource `Id`
- **`from`** (`osid.calendaring.DateTime`) – starting date
- **`to`** (`osid.calendaring.DateTime`) – ending date

Returns the returned `Proficiency` list

Return type `osid.learning.ProficiencyList`

Raise `InvalidArgument` – `from` is greater than `to`

Raise `NullArgument` – `resource_id, from` or `to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_proficiencies_by_genus_type_for_resource`**(*resource_id*, *proficiency_genus_type*)

Gets a `ProficiencyList` relating to the given resource and proficiency genus `Type`.

Parameters

- **`resource_id`** (`osid.id.Id`) – a resource `Id`
- **`proficiency_genus_type`** (`osid.type.Type`) – a proficiency genus type

Returns the returned `Proficiency` list

Return type `osid.learning.ProficiencyList`

Raise `NullArgument` – `resource_id` or `proficiency_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_proficiencies_by_genus_type_for_resource_on_date`**(*resource_id, proficiency_genus_type, from_, to*)

Gets a `ProficiencyList` of the given proficiency genus type relating to the given resource effective during the entire given date range inclusive but not confined to the date range.

Parameters

- **`resource_id`** (`osid.id.Id`) – a resource `Id`
- **`proficiency_genus_type`** (`osid.type.Type`) – a proficiency genus type
- **`from`** (`osid.calendaring.DateTime`) – starting date
- **`to`** (`osid.calendaring.DateTime`) – ending date

Returns the returned `Proficiency` list

Return type `osid.learning.ProficiencyList`

Raise `InvalidArgument` – `from` is greater than `to`

Raise `NullArgument` – `resource_id, proficiency_genus_type, from` or `to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_proficiencies_for_resources`**(*resource_ids*)

Gets a `ProficiencyList` relating to the given resources.

Parameters **`resource_ids`** (`osid.id.IdList`) – the resource `Ids`

Returns the returned `Proficiency` list

Return type `osid.learning.ProficiencyList`

Raise `NullArgument` – `resource_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_proficiencies_for_objective_and_resource`**(*objective_id, resource_id*)

Gets a `ProficiencyList` relating to the given objective and resource ````.

Parameters

- **`objective_id`** (`osid.id.Id`) – an objective `Id`

- **resource_id** (osid.id.Id) – a resource Id

Returns the returned Proficiency list

Return type osid.learning.ProficiencyList

Raise NullArgument – objective_id or resource_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

ObjectiveBank.**get_proficiencies_for_objective_and_resource_on_date**(*objective_id, resource_id, from_, to*)

Gets a ProficiencyList relating to the given resource and objective effective during the entire given date range inclusive but not confined to the date range.

Parameters

- **objective_id** (osid.id.Id) – an objective Id
- **resource_id** (osid.id.Id) – a resource Id
- **from** (osid.calendaring.DateTime) – starting date
- **to** (osid.calendaring.DateTime) – ending date

Returns the returned Proficiency list

Return type osid.learning.ProficiencyList

Raise InvalidArgument – from is greater than to

Raise NullArgument – objective_id, resource_id, from or to is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

ObjectiveBank.**get_proficiencies_by_genus_type_for_objective_and_resource**(*objective_id, resource_id, proficiency_genus_type*)

Gets a ProficiencyList of the given genus type relating to the given objective and resource ‘‘‘‘.

Parameters

- **objective_id** (osid.id.Id) – an objective Id
- **resource_id** (osid.id.Id) – a resource Id
- **proficiency_genus_type** (osid.type.Type) – a proficiency genus type

Returns the returned Proficiency list

Return type osid.learning.ProficiencyList

Raise `NullArgument` – `objective_id, resource_id` or `proficiency_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_proficiencies_by_genus_type_for_objective_and_resource_on_date`**`(`*objectiv re-source_ pro-fi-ciency_ from_, to*`)`

Gets a `ProficiencyList` of the given genus type relating to the given resource and objective effective during the entire given date range inclusive but not confined to the date range.

Parameters

- **`objective_id`** (`osid.id.Id`) – an objective `Id`
- **`resource_id`** (`osid.id.Id`) – a resource `Id`
- **`proficiency_genus_type`** (`osid.type.Type`) – a proficiency genus type
- **`from`** (`osid.calendaring.DateTime`) – starting date
- **`to`** (`osid.calendaring.DateTime`) – ending date

Returns the returned `Proficiency` list

Return type `osid.learning.ProficiencyList`

Raise `InvalidArgument` – `from` is greater than `to`

Raise `NullArgument` – `objective_id, resource_id, proficiency_genus_type, from` or `to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`proficiencies`**

Gets all `Proficiencies`.

Returns a list of `Proficiencies`

Return type `osid.learning.ProficiencyList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Proficiency Query Methods

`ObjectiveBank.`**`objective_bank_id`**

Gets the `ObjectiveBank Id` associated with this session.

Returns the `ObjectiveBank Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`objective_bank`**

Gets the `ObjectiveBank` associated with this session.

Returns the obective bank

Return type `osid.learning.ObjectiveBank`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`can_search_proficiencies`**`()`

Tests if this user can perform `Proficiency` lookups.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may not offer lookup operations to unauthorized users.

Returns `false` if search methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`use_federated_objective_bank_view`**`()`

Federates the view for methods in this session.

A federated view will include proficiencies in objective banks which are children of this objective bank in the obective bank hierarchy.

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`use_isolated_objective_bank_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts lookups to this objective bank only.

compliance: mandatory – This method is must be implemented.

`ObjectiveBank.`**`proficiency_query`**

Gets a proficiency query.

Returns the proficiency query

Return type `osid.learning.ProficiencyQuery`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_proficiencies_by_query`**`(`*proficiency_query*`)`

Gets a list of `Proficiencies` matching the given proficiency query.

Parameters **`proficiency_query`** (`osid.learning.ProficiencyQuery`) – the proficiency query

Returns the returned `ProficiencyList`

Return type `osid.learning.ProficiencyList`

Raise `NullArgument` – `proficiency_query` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `proficiency_query` is not of this service

compliance: mandatory – This method must be implemented.

Proficiency Admin Methods

`ObjectiveBank.`**`objective_bank_id`**

Gets the `ObjectiveBank Id` associated with this session.

Returns the `ObjectiveBank Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`objective_bank`**

Gets the `ObjectiveBank` associated with this session.

Returns the obective bank

Return type `osid.learning.ObjectiveBank`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`can_create_proficiencies`**`()`

Tests if this user can create `Proficiencies`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known creating a `Proficiency` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer create operations to an unauthorized user.

Returns `false` if `Proficiency` creation is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`can_create_proficiency_with_record_types`**`(`*proficiency_record_types*`)`

Tests if this user can create a single `Proficiency` using the desired record types.

While `LearningManager.getProficiencyRecordTypes()` can be used to examine which records are supported, this method tests which record(s) are required for creating a specific `Proficiency`. Providing an empty array tests if a `Proficiency` can be created with no records.

Parameters **`proficiency_record_types`** (`osid.type.Type[]`) – array of proficiency record types

Returns `true` if `Proficiency` creation using the specified record `Types` is supported, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `proficiency_record_types` is `null`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_proficiency_form_for_create`**(*objective_id*, *resource_id*, *proficiency_record_types*)

Gets the proficiency form for creating new proficiencies.

A new form should be requested for each create transaction.

Parameters

- **`objective_id`** (`osid.id.Id`) – the `Id` of the `Objective`
- **`resource_id`** (`osid.id.Id`) – the `Id` of the `Resource`
- **`proficiency_record_types`** (`osid.type.Type[]`) – array of proficiency record types

Returns the proficiency form

Return type `osid.learning.ProficiencyForm`

Raise `NotFound` – `objective_id` or `resource_id` is not found

Raise `NullArgument` – `objective_id, resource_id,` or `proficieny_record_types` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – unable to get form for requested record types

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`create_proficiency`**(*proficiency_form*)

Creates a new `Proficiency`.

A new form should be requested for each create transaction.

Parameters **`proficiency_form`** (`osid.learning.ProficiencyForm`) – the form for this `Proficiency`

Returns the new `Proficiency`

Return type `osid.learning.Proficiency`

Raise `IllegalState` – `proficiency_form` already used in a create transaction

Raise `InvalidArgument` – one or more of the form elements is invalid

Raise `NullArgument` – `proficiency_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `proficiency_form` did not originate from `get_proficiency_form_for_create()`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`can_update_proficiencies`**()

Tests if this user can update `Proficiencies`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating a `Proficiency` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer update operations to an unauthorized user.

Returns `false` if `Proficiency` modification is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`get_proficiency_form_for_update`**`(`*proficiency_id*`)`

Gets the proficiency form for updating an existing proficiency.

Parameters **`proficiency_id`** (`osid.id.Id`) – the `Id` of the `Proficiency`

Returns the proficiency form

Return type `osid.learning.ProficiencyForm`

Raise `NotFound` – `proficiency_id` is not found

Raise `NullArgument` – `proficiency_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`update_proficiency`**`(`*proficiency_form*`)`

Updates an existing proficiency.

Parameters **`proficiency_form`** (`osid.learning.ProficiencyForm`) – the form containing the elements to be updated

Raise `IllegalState` – `proficiency_form` already used in an update transaction

Raise `InvalidArgument` – the form contains an invalid value

Raise `NullArgument` – `proficiency_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `proficiency_form` did not originate from `get_proficiency_form_for_update()`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`can_delete_proficiencies`**`()`

Tests if this user can delete `Proficiencies`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting a `Proficiency` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer delete operations to an unauthorized user.

Returns `false` if `Proficiency` deletion is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`delete_proficiency`**`(`*proficiency_id*`)`

Deletes a `Proficiency`.

Parameters **`proficiency_id`** (`osid.id.Id`) – the `Id` of the `Proficiency` to remove

Raise `NotFound` – `proficiency_id` not found

Raise `NullArgument` – `proficiency_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`delete_proficiencies`**`()`

Deletes all proficiencies in this `ObjectiveBank`.

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`can_manage_proficiency_aliases`**`()`

Tests if this user can manage `Id` aliases for proficiency entries.

A return of true does not guarantee successful authorization. A return of false indicates that it is known changing an alias will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer alias operations to an unauthorized user.

Returns `false` if `Proficiency` aliasing is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ObjectiveBank.`**`alias_proficiency`**`(`*proficiency_id*, *alias_id*`)`

Adds an `Id` to a `Proficiency` for the purpose of creating compatibility.

The primary `Id` of the `Proficiency` is determined by the provider. The new `Id` performs as an alias to the primary `Id`. If the alias is a pointer to another proficiency, it is reassigned to the given proficiency `Id`.

Parameters

- **`proficiency_id`** (`osid.id.Id`) – the `Id` of a `Proficiency`
- **`alias_id`** (`osid.id.Id`) – the alias `Id`

Raise `AlreadyExists` – `alias_id` is already assigned

Raise `NotFound` – `proficiency_id` not found

Raise `NullArgument` – `proficiency_id` or `alias_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Objects

Objective

class `dlkit.learning.objects.`**`Objective`**

Bases: *`dlkit.osid.objects.OsidObject`*, *`dlkit.osid.markers.Federateable`*

An `Objective` is a statable learning objective.

`has_assessment``()`

Tests if an assessment is associated with this objective.

Returns `true` if an assessment exists, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

assessment_id
Gets the assessment `Id` associated with this learning objective.

Returns the assessment `Id`

Return type `osid.id.Id`

Raise `IllegalState` – `has_assessment()` is `false`

compliance: mandatory – This method must be implemented.

assessment
Gets the assessment associated with this learning objective.

Returns the assessment

Return type `osid.assessment.Assessment`

Raise `IllegalState` – `has_assessment()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

has_knowledge_category()
Tests if this objective has a knowledge dimension.

Returns `true` if a knowledge category exists, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

knowledge_category_id
Gets the grade `Id` associated with the knowledge dimension.

Returns the grade `Id`

Return type `osid.id.Id`

Raise `IllegalState` – `has_knowledge_category()` is `false`

compliance: mandatory – This method must be implemented.

knowledge_category
Gets the grade associated with the knowledge dimension.

Returns the grade

Return type `osid.grading.Grade`

Raise `IllegalState` – `has_knowledge_category()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

has_cognitive_process()
Tests if this objective has a cognitive process type.

Returns `true` if a cognitive process exists, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

cognitive_process_id

Gets the grade Id associated with the cognitive process.

Returns the grade Id

Return type osid.id.Id

Raise IllegalState – has_cognitive_process() is false

compliance: mandatory – This method must be implemented.

cognitive_process

Gets the grade associated with the cognitive process.

Returns the grade

Return type osid.grading.Grade

Raise IllegalState – has_cognitive_process() is false

Raise OperationFailed – unable to complete request

compliance: mandatory – This method must be implemented.

get_objective_record(*objective_record_type*)

Gets the objective bank record corresponding to the given Objective record Type.

This method is used to retrieve an object implementing the requested record. The objective_record_type may be the Type returned in get_record_types() or any of its parents in a Type hierarchy where has_record_type(objective_record_type) is true .

Parameters **objective_record_type** (osid.type.Type) – an objective record type

Returns the objective record

Return type osid.learning.records.ObjectiveRecord

Raise NullArgument – objective_record_type is null

Raise OperationFailed – unable to complete request

Raise Unsupported – has_record_type(objective_record_type) is false

compliance: mandatory – This method must be implemented.

Objective Form

class dlkit.learning.objects.**ObjectiveForm**

Bases: *dlkit.osid.objects.OsidObjectForm*, *dlkit.osid.objects.OsidFederateableForm*

This is the form for creating and updating Objectives.

Like all OsidForm objects, various data elements may be set here for use in the create and update methods in the ObjectiveAdminSession. For each data element that may be set, metadata may be examined to provide display hints or data constraints.

assessment_metadata

Gets the metadata for an assessment.

Returns metadata for the assessment

Return type osid.Metadata

compliance: mandatory – This method must be implemented.

assessment

knowledge_category_metadata

Gets the metadata for a knowledge category.

Returns metadata for the knowledge category

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

knowledge_category

cognitive_process_metadata

Gets the metadata for a cognitive process.

Returns metadata for the cognitive process

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

cognitive_process

get_objective_form_record(*objective_record_type*)

Gets the `ObjectiveFormRecord` corresponding to the given objective record `Type`.

Parameters **objective_record_type** (`osid.type.Type`) – the objective record type

Returns the objective form record

Return type `osid.learning.records.ObjectiveFormRecord`

Raise `NullArgument` – `objective_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(objective_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Objective List

class `dlkit.learning.objects.`**ObjectiveList**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists`, `ObjectiveList` provides a means for accessing `Objective` elements sequentially either one at a time or many at a time.

Examples: while (ol.hasNext()) { Objective objective = ol.getNextObjective(); }

or

while (ol.hasNext()) { Objective[] objectives = ol.getNextObjectives(ol.available());

}

next_objective

Gets the next `Objective` in this list.

Returns the next `Objective` in this list. The `has_next()` method should be used to test that a next `Objective` is available before calling this method.

Return type `osid.learning.Objective`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_objectives`(*n*)

Gets the next set of `Objective` elements in this list which must be less than or equal to the number returned from `available()`.

Parameters **`n`** (`cardinal`) – the number of `Objective` elements requested which should be less than or equal to `available()`

Returns an array of `Objective` elements.The length of the array is less than or equal to the number specified.

Return type `osid.learning.Objective`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Objective Node

class `dlkit.learning.objects.`**`ObjectiveNode`**

Bases: *`dlkit.osid.objects.OsidNode`*

This interface is a container for a partial hierarchy retrieval.

The number of hierarchy levels traversable through this interface depend on the number of levels requested in the `ObjectiveHierarchySession`.

`objective`

Gets the `Objective` at this node.

Returns the objective represented by this node

Return type `osid.learning.Objective`

compliance: mandatory – This method must be implemented.

`parent_objective_nodes`

Gets the parents of this objective.

Returns the parents of the `id`

Return type `osid.learning.ObjectiveNodeList`

compliance: mandatory – This method must be implemented.

`child_objective_nodes`

Gets the children of this objective.

Returns the children of this objective

Return type `osid.learning.ObjectiveNodeList`

compliance: mandatory – This method must be implemented.

Objective Node List

class `dlkit.learning.objects.`**`ObjectiveNodeList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists`, `ObjectiveNodeList` provides a means for accessing `ObjectiveNode` elements sequentially either one at a time or many at a time.

Examples: while (onl.hasNext()) { ObjectiveNode node = onl.getNextObjectiveNode(); }

or

while (onl.hasNext()) { ObjectiveNode[] nodes = onl.getNextObjectiveNodes(onl.available());

}

`next_objective_node`

Gets the next `ObjectiveNode` in this list.

Returns the next `ObjectiveNode` in this list. The `has_next()` method should be used to test that a next `ObjectiveNode` is available before calling this method.

Return type `osid.learning.ObjectiveNode`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_objective_nodes`(*n*)

Gets the next set of `ObjectiveNode` elements in this list which must be less than or equal to the number returned from `available()`.

Parameters n (`cardinal`) – the number of `ObjectiveNode` elements requested which should be less than or equal to `available()`

Returns an array of `ObjectiveNode` elements.The length of the array is less than or equal to the number specified.

Return type `osid.learning.ObjectiveNode`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Activity

class `dlkit.learning.objects.`**`Activity`**

Bases: *`dlkit.osid.objects.OsidObject`*, *`dlkit.osid.markers.Subjugateable`*

An `Activity` represents learning material or other learning activities to meet an objective.

An Activity has may relate to a set of `Asssts` for self learning, recommended `Courses` to take, or a learning `Assessment`. The learning `Assessment` differs from the `Objective Assessment` in that the latter used to test for proficiency in the `Objective`.

Generally, an `Activity` should focus on one of assets, courses, assessments, or some other specific activity related to the objective described or related in the `ActivityRecord`.

`objective_id`

Gets the `Id` of the related objective.

Returns the objective `Id`

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

objective

Gets the related objective.

Returns the related objective

Return type `osid.learning.Objective`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

is_asset_based_activity()

Tests if this is an asset based activity.

Returns `true` if this activity is based on assets, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

asset_ids

Gets the `Ids` of any assets associated with this activity.

Returns list of asset `Ids`

Return type `osid.id.IdList`

Raise `IllegalState` – `is_asset_based_activity()` is `false`

compliance: mandatory – This method must be implemented.

assets

Gets any assets associated with this activity.

Returns list of assets

Return type `osid.repository.AssetList`

Raise `IllegalState` – `is_asset_based_activity()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

is_course_based_activity()

Tests if this is a course based activity.

Returns `true` if this activity is based on courses, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

course_ids

Gets the `Ids` of any courses associated with this activity.

Returns list of course `Ids`

Return type `osid.id.IdList`

Raise `IllegalState` – `is_course_based_activity()` is `false`

compliance: mandatory – This method must be implemented.

courses

Gets any courses associated with this activity.

Returns list of courses

Return type `osid.course.CourseList`

Raise `IllegalState` – `is_course_based_activity()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`is_assessment_based_activity`()

Tests if this is an assessment based activity.

These assessments are for learning the objective and not for assessing prodiciency in the objective.

Returns `true` if this activity is based on assessments, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`assessment_ids`

Gets the `Ids` of any assessments associated with this activity.

Returns list of assessment `Ids`

Return type `osid.id.IdList`

Raise `IllegalState` – `is_assessment_based_activity()` is `false`

compliance: mandatory – This method must be implemented.

`assessments`

Gets any assessments associated with this activity.

Returns list of assessments

Return type `osid.assessment.AssessmentList`

Raise `IllegalState` – `is_assessment_based_activity()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_activity_record`(*activity_record_type*)

Gets the activity record corresponding to the given `Activity` record `Type`.

This method is used to retrieve an object implementing the requested record. The `activity_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(activity_record_type)` is `true` .

Parameters **`activity_record_type`** (`osid.type.Type`) – the type of the record to retrieve

Returns the activity record

Return type `osid.learning.records.ActivityRecord`

Raise `NullArgument` – `activity_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(activity_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Activity Form

class `dlkit.learning.objects.`**`ActivityForm`**

Bases: *`dlkit.osid.objects.OsidObjectForm`*, *`dlkit.osid.objects.OsidSubjugateableForm`*

This is the form for creating and updating `Activities`.

Like all `OsidForm` objects, various data elements may be set here for use in the create and update methods in the `ActivityAdminSession`. For each data element that may be set, metadata may be examined to provide display hints or data constraints.

`assets_metadata`

Gets the metadata for the assets.

Returns metadata for the assets

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`assets`

`courses_metadata`

Gets the metadata for the courses.

Returns metadata for the courses

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`courses`

`assessments_metadata`

Gets the metadata for the assessments.

Returns metadata for the assessments

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`assessments`

`get_activity_form_record`(*activity_record_type*)

Gets the `ActivityFormRecord` corresponding to the given activity record `Type`.

Parameters **`activity_record_type`** (`osid.type.Type`) – the activity record type

Returns the activity form record

Return type `osid.learning.records.ActivityFormRecord`

Raise `NullArgument` – `activity_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(activity_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Activity List

class `dlkit.learning.objects.`**`ActivityList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists, ActivityList` provides a means for accessing `Activity` elements sequentially either one at a time or many at a time.

Examples: while (al.hasNext()) { Activity activity = al.getNextActivity(); }

or

while (al.hasNext()) { Activity[] activities = al.getNextActivities(al.available());

}

`next_activity`

Gets the next `Activity` in this list.

Returns the next `Activity` in this list. The `has_next()` method should be used to test that a next `Activity` is available before calling this method.

Return type `osid.learning.Activity`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_activities`(*n*)

Gets the next set of `Activity` elements in this list which must be less than or equal to the number returned from `available()`.

Parameters **n** (`cardinal`) – the number of `Activity` elements requested which should be less than or equal to `available()`

Returns an array of `Activity` elements.The length of the array is less than or equal to the number specified.

Return type `osid.learning.Activity`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Proficiency

class `dlkit.learning.objects.`**`Proficiency`**

Bases: *`dlkit.osid.objects.OsidRelationship`*

A `Proficiency` represents a competency of a leraning objective.

`resource_id`

Gets the resource `Id` to whom this proficiency applies.

Returns the resource `Id`

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

resource
Gets the resource to whom this proficiency applies.

Returns the resource

Return type `osid.resource.Resource`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

objective_id
Gets the objective `Id` to whom this proficiency applies.

Returns the objective `Id`

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

objective
Gets the objective to whom this proficiency applies.

Returns the objective

Return type `osid.learning.Objective`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

completion
Gets the completion of this objective as a percentage 0-100.

Returns the completion

Return type `decimal`

compliance: mandatory – This method must be implemented.

has_level()
Tests if a proficiency level is available.

Returns `true` if a level is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

level_id
Gets the proficiency level expressed as a grade.

Returns the grade `Id`

Return type `osid.id.Id`

Raise `IllegalState` – `has_level()` is `false`

compliance: mandatory – This method must be implemented.

level
Gets the proficiency level expressed as a grade.

Returns the grade

Return type `osid.grading.Grade`

Raise `IllegalState` – `has_level()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_proficiency_record`(*proficiency_record_type*)

Gets the proficiency record corresponding to the given `Proficiency` record `Type`.

This method is used to retrieve an object implementing the requested record. The `proficiency_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(proficiency_record_type)` is `true` .

Parameters **`proficiency_record_type`** (`osid.type.Type`) – the type of proficiency record to retrieve

Returns the proficiency record

Return type `osid.learning.records.ProficiencyRecord`

Raise `NullArgument` – `proficiency_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(proficiency_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Proficiency Form

class `dlkit.learning.objects.`**`ProficiencyForm`**

Bases: *`dlkit.osid.objects.OsidRelationshipForm`*

This is the form for creating and updating `Proficiencies`.

Like all `OsidForm` objects, various data elements may be set here for use in the create and update methods in the `ProficiencyAdminSession`. For each data element that may be set, metadata may be examined to provide display hints or data constraints.

`completion_metadata`

Gets the metadata for completion percentage.

Returns metadata for the completion percentage

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`completion`

`level_metadata`

Gets the metadata for a level.

Returns metadata for the grade level

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`level`

`get_proficiency_form_record`(*proficiency_record_type*)

Gets the `ProficiencyFormRecord` corresponding to the given proficiency record `Type`.

Parameters **`proficiency_record_type`** (`osid.type.Type`) – a proficiency record type

Returns the proficiency form record

Return type `osid.learning.records.ProficiencyFormRecord`

Raise `NullArgument` – `proficiency_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(proficiency_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Proficiency List

class `dlkit.learning.objects.`**`ProficiencyList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists, ProficiencyList` provides a means for accessing `Proficiency` elements sequentially either one at a time or many at a time.

Examples: while (pl.hasNext()) { Proficiency proficiency = pl.getNextProficiency(); }

or

while (pl.hasNext()) { Proficiency[] proficiencies = pl.getNextProficiencies(pl.available());

}

`next_proficiency`

Gets the next `Proficiency` in this list.

Returns the next `Proficiency` in this list. The `has_next()` method should be used to test that a next `Proficiency` is available before calling this method.

Return type `osid.learning.Proficiency`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_proficiencies``(`*n*`)`

Gets the next set of `Proficiency` elements in this list.

The specified amount must be less than or equal to the return from `available()`.

Parameters **n** (`cardinal`) – the number of `Proficiency` elements requested which must be less than or equal to `available()`

Returns an array of `Proficiency` elements.The length of the array is less than or equal to the number specified.

Return type `osid.learning.Proficiency`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Objective Bank

class `dlkit.learning.objects.`**`ObjectiveBank`**(*abc_learning_objects.ObjectiveBank, osid_objects.OsidCatalog*)
`:noindex:`

`get_objective_bank_record`(*objective_bank_record_type*)
Gets the objective bank record corresponding to the given `ObjectiveBank` record `Type`.

This method is used to retrieve an object implementing the requested record. The `objective_bank_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(objective_bank_record_type)` is `true` .

Parameters **`objective_bank_record_type`** (`osid.type.Type`) – an objective bank record type

Returns the objective bank record

Return type `osid.learning.records.ObjectiveBankRecord`

Raise `NullArgument` – `objective_bank_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(objective_bank_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Objective Bank Form

class `dlkit.learning.objects.`**`ObjectiveBankForm`**
Bases: *`dlkit.osid.objects.OsidCatalogForm`*

This is the form for creating and updating objective banks.

Like all `OsidForm` objects, various data elements may be set here for use in the create and update methods in the `ObjectiveBankAdminSession`. For each data element that may be set, metadata may be examined to provide display hints or data constraints.

`get_objective_bank_form_record`(*objective_bank_record_type*)
Gets the `ObjectiveBankFormRecord` corresponding to the given objective bank record `Type`.

Parameters **`objective_bank_record_type`** (`osid.type.Type`) – an objective bank record type

Returns the objective bank form record

Return type `osid.learning.records.ObjectiveBankFormRecord`

Raise `NullArgument` – `objective_bank_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(objective_bank_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Objective Bank List

class `dlkit.learning.objects.`**`ObjectiveBankList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists, ObjectiveBankList` provides a means for accessing `ObjectiveBank` elements sequentially either one at a time or many at a time.

Examples: while (obl.hasNext()) { ObjectiveBank objectiveBanks = obl.getNextObjectiveBank(); }

or

while (obl.hasNext()) { ObjectiveBank[] objectivBanks = obl.getNextObjectiveBanks(obl.available());

}

`next_objective_bank`

Gets the next `ObjectiveBank` in this list.

Returns the next `ObjectiveBank` in this list. The `has_next()` method should be used to test that a next `ObjectiveBank` is available before calling this method.

Return type `osid.learning.ObjectiveBank`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_objective_banks`(*n*)

Gets the next set of `ObjectiveBank` elements in this list which must be less than or equal to the return from `available()`.

Parameters n (`cardinal`) – the number of `ObjectiveBank` elements requested which must be less than or equal to `available()`

Returns an array of `ObjectiveBank` elements.The length of the array is less than or equal to the number specified.

Return type `osid.learning.ObjectiveBank`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Objective Bank Node

class `dlkit.learning.objects.`**`ObjectiveBankNode`**

Bases: *`dlkit.osid.objects.OsidNode`*

This interface is a container for a partial hierarchy retrieval.

The number of hierarchy levels traversable through this interface depend on the number of levels requested in the `ObjectiveBankHierarchySession`.

`objective_bank`

Gets the `ObjectiveBank` at this node.

Returns the objective bank represented by this node

Return type `osid.learning.ObjectiveBank`

compliance: mandatory – This method must be implemented.

parent_objective_bank_nodes

Gets the parents of this objective bank.

Returns the parents of the `id`

Return type `osid.learning.ObjectiveBankNodeList`

compliance: mandatory – This method must be implemented.

child_objective_bank_nodes

Gets the children of this objective bank.

Returns the children of this objective bank

Return type `osid.learning.ObjectiveBankNodeList`

compliance: mandatory – This method must be implemented.

Objective Bank Node List

class `dlkit.learning.objects.`**`ObjectiveBankNodeList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists`, `ObjectiveBankNodeList` provides a means for accessing `ObjectiveBankNode` elements sequentially either one at a time or many at a time.

Examples: while (obnl.hasNext()) { ObjectiveBankNode node bank = obnl.getNextObjectiveBankNode(); }

or

while (obnl.hasNext()) { ObjectiveBankNode[] nodes = obnl.getNextObjectiveBankNodes(obnl.available());

}

next_objective_bank_node

Gets the next `ObjectiveBankNode` in this list.

Returns the next `ObjectiveBankNode` in this list. The `has_next()` method should be used to test that a next `ObjectiveBankNode` is available before calling this method.

Return type `osid.learning.ObjectiveBankNode`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

get_next_objective_bank_nodes(*n*)

Gets the next set of `ObjectiveBankNode` elements in this list which must be less than or equal to the return from `available()`.

Parameters **n** (`cardinal`) – the number of `ObjectiveBankNode` elements requested which must be less than or equal to `available()`

Returns an array of `ObjectiveBankNode` elements.The length of the array is less than or equal to the number specified.

Return type `osid.learning.ObjectiveBankNode`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Queries

Objective Query

class `dlkit.learning.queries.`**`ObjectiveQuery`**

Bases: *`dlkit.osid.queries.OsidObjectQuery`*, *`dlkit.osid.queries.OsidFederateableQuery`*

This is the query for searching objectives.

Each method match request produces an `AND` term while multiple invocations of a method produces a nested `OR`.

`match_assessment_id`(*assessment_id*, *match*)

Sets the assessment `Id` for this query.

Parameters

- **`assessment_id`** (`osid.id.Id`) – an assessment `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `assessment_id` is `null`

compliance: mandatory – This method must be implemented.

`assessment_id_terms`

`supports_assessment_query`()

Tests if an `AssessmentQuery` is available for querying activities.

Returns `true` if an assessment query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`assessment_query`

Gets the query for an assessment.

Multiple retrievals produce a nested `OR` term.

Returns the assessment query

Return type `osid.assessment.AssessmentQuery`

Raise `Unimplemented` – `supports_assessment_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_assessment_query()‘‘ is ‘‘true‘‘.

`match_any_assessment`(*match*)

Matches an objective that has any assessment assigned.

Parameters **`match`** (`boolean`) – `true` to match objectives with any assessment, `false` to match objectives with no assessment

compliance: mandatory – This method must be implemented.

`assessment_terms`

`match_knowledge_category_id`(*grade_id*, *match*)

Sets the knowledge category `Id` for this query.

Parameters

- **grade_id** (osid.id.Id) – a grade Id
- **match** (boolean) – true for a positive match, false for a negative match

Raise NullArgument – grade_id is null

compliance: mandatory – This method must be implemented.

knowledge_category_id_terms

supports_knowledge_category_query()
Tests if a GradeQuery is available for querying knowledge categories.

Returns true if a grade query is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

knowledge_category_query
Gets the query for a knowledge category.

Multiple retrievals produce a nested OR term.

Returns the grade query

Return type osid.grading.GradeQuery

Raise Unimplemented – supports_knowledge_category_query() is false

compliance: optional – This method must be implemented if ‘‘supports_knowledge_category_query()‘‘ is ‘‘true‘‘.

match_any_knowledge_category(*match*)
Matches an objective that has any knowledge category.

Parameters match (boolean) – true to match objectives with any knowledge category, false to match objectives with no knowledge category

compliance: mandatory – This method must be implemented.

knowledge_category_terms

match_cognitive_process_id(*grade_id*, *match*)
Sets the cognitive process Id for this query.

Parameters

- **grade_id** (osid.id.Id) – a grade Id
- **match** (boolean) – true for a positive match, false for a negative match

Raise NullArgument – grade_id is null

compliance: mandatory – This method must be implemented.

cognitive_process_id_terms

supports_cognitive_process_query()
Tests if a GradeQuery is available for querying cognitive processes.

Returns true if a grade query is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

`cognitive_process_query`

Gets the query for a cognitive process.

Multiple retrievals produce a nested `OR` term.

Returns the grade query

Return type `osid.grading.GradeQuery`

Raise `Unimplemented` – `supports_cognitive_process_query()` is `false`

compliance: optional – This method must be implemented if ``supports_cognitive_process_query()`` is ``true``.

`match_any_cognitive_process`(*match*)

Matches an objective that has any cognitive process.

Parameters **`match`** (`boolean`) – `true` to match objectives with any cognitive process, `false` to match objectives with no cognitive process

compliance: mandatory – This method must be implemented.

`cognitive_process_terms`

`match_activity_id`(*activity_id*, *match*)

Sets the activity `Id` for this query.

Parameters

- **`activity_id`** (`osid.id.Id`) – an activity `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `activity_id` is `null`

compliance: mandatory – This method must be implemented.

`activity_id_terms`

`supports_activity_query`()

Tests if an `ActivityQuery` is available for querying activities.

Returns `true` if an activity query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`activity_query`

Gets the query for an activity.

Multiple retrievals produce a nested `OR` term.

Returns the activity query

Return type `osid.learning.ActivityQuery`

Raise `Unimplemented` – `supports_activity_query()` is `false`

compliance: optional – This method must be implemented if ``supports_activity_query()`` is ``true``.

`match_any_activity`(*match*)

Matches an objective that has any related activity.

Parameters **`match`** (`boolean`) – `true` to match objectives with any activity, `false` to match objectives with no activity

compliance: mandatory – This method must be implemented.

activity_terms

match_requisite_objective_id(*requisite_objective_id*, *match*)
Sets the requisite objective `Id` for this query.

Parameters

- **requisite_objective_id** (`osid.id.Id`) – a requisite objective `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `requisite_objective_id` is `null`

compliance: mandatory – This method must be implemented.

requisite_objective_id_terms

supports_requisite_objective_query()
Tests if an `ObjectiveQuery` is available for querying requisite objectives.

Returns `true` if an objective query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

requisite_objective_query
Gets the query for a requisite objective.

Multiple retrievals produce a nested `OR` term.

Returns the objective query

Return type `osid.learning.ObjectiveQuery`

Raise `Unimplemented` – `supports_requisite_objective_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_requisite_objective_query()‘‘ is ‘‘true‘‘.

match_any_requisite_objective(*match*)
Matches an objective that has any related requisite.

Parameters **match** (`boolean`) – `true` to match objectives with any requisite, `false` to match objectives with no requisite

compliance: mandatory – This method must be implemented.

requisite_objective_terms

match_dependent_objective_id(*dependent_objective_id*, *match*)
Sets the dependent objective `Id` to query objectives dependent on the given objective.

Parameters

- **dependent_objective_id** (`osid.id.Id`) – a dependent objective `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `dependent_objective_id` is `null`

compliance: mandatory – This method must be implemented.

dependent_objective_id_terms

supports_depndent_objective_query()
Tests if an `ObjectiveQuery` is available for querying dependent objectives.

Returns `true` if an objective query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`dependent_objective_query`

Gets the query for a dependent objective.

Multiple retrievals produce a nested `OR` term.

Returns the objective query

Return type `osid.learning.ObjectiveQuery`

Raise `Unimplemented` – `supports_dependent_objective_query()` is `false`

compliance: optional – This method must be implemented if ``supports_dependent_query()`` is ``true``.

`match_any_dependent_objective`(*match*)

Matches an objective that has any related dependents.

Parameters **`match`** (`boolean`) – `true` to match objectives with any dependent, `false` to match objectives with no dependents

compliance: mandatory – This method must be implemented.

`dependent_objective_terms`

`match_equivalent_objective_id`(*equivalent_objective_id*, *match*)

Sets the equivalent objective `Id` to query equivalents.

Parameters

- **`equivalent_objective_id`** (`osid.id.Id`) – an equivalent objective `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `equivalent_objective_id` is `null`

compliance: mandatory – This method must be implemented.

`equivalent_objective_id_terms`

`supports_equivalent_objective_query`()

Tests if an `ObjectiveQuery` is available for querying equivalent objectives.

Returns `true` if an objective query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`equivalent_objective_query`

Gets the query for an equivalent objective.

Multiple retrievals produce a nested `OR` term.

Returns the objective query

Return type `osid.learning.ObjectiveQuery`

Raise `Unimplemented` – `supports_equivalent_objective_query()` is `false`

compliance: optional – This method must be implemented if ``supports_equivalent_query()`` is ``true``.

`match_any_equivalent_objective`(*match*)

Matches an objective that has any related equivalents.

Parameters **`match`** (`boolean`) – `true` to match objectives with any equivalent, `false` to match objectives with no equivalents

compliance: mandatory – This method must be implemented.

equivalent_objective_terms

match_ancestor_objective_id(*objective_id*, *match*)

Sets the objective `Id` for this query to match objectives that have the specified objective as an ancestor.

Parameters

- **objective_id** (`osid.id.Id`) – an objective `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `objective_id` is `null`

compliance: mandatory – This method must be implemented.

ancestor_objective_id_terms

supports_ancestor_objective_query()

Tests if an `ObjectiveQuery` is available.

Returns `true` if an objective query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

ancestor_objective_query

Gets the query for an objective.

Multiple retrievals produce a nested `OR` term.

Returns the objective query

Return type `osid.learning.ObjectiveQuery`

Raise `Unimplemented` – `supports_ancestor_objective_query()` is `false`

compliance: optional – This method must be implemented if ``supports_ancestor_objective_query()`` is ``true``.

match_any_ancestor_objective(*match*)

Matches objectives that have any ancestor.

Parameters **match** (`boolean`) – `true` to match objective with any ancestor, `false` to match root objectives

compliance: mandatory – This method must be implemented.

ancestor_objective_terms

match_descendant_objective_id(*objective_id*, *match*)

Sets the objective `Id` for this query to match objectives that have the specified objective as a descendant.

Parameters

- **objective_id** (`osid.id.Id`) – an objective `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `objective_id` is `null`

compliance: mandatory – This method must be implemented.

descendant_objective_id_terms

supports_descendant_objective_query()

Tests if an `ObjectiveQuery` is available.

Returns `true` if an objective query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`descendant_objective_query`
Gets the query for an objective.

Multiple retrievals produce a nested `OR` term.

Returns the objective query

Return type `osid.learning.ObjectiveQuery`

Raise `Unimplemented` – `supports_descendant_objective_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_descendant_objective_query()‘‘ is ‘‘true‘‘.

`match_any_descendant_objective`(*match*)
Matches objectives that have any ancestor.

Parameters **`match`** (`boolean`) – `true` to match objectives with any ancestor, `false` to match leaf objectives

compliance: mandatory – This method must be implemented.

`descendant_objective_terms`

`match_objective_bank_id`(*objective_bank_id*, *match*)
Sets the objective bank `Id` for this query.

Parameters

- **`objective_bank_id`** (`osid.id.Id`) – an objective bank `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `objective_bank_id` is `null`

compliance: mandatory – This method must be implemented.

`objective_bank_id_terms`

`supports_objective_bank_query`()
Tests if a `ObjectiveBankQuery` is available for querying objective banks.

Returns `true` if an objective bank query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`objective_bank_query`
Gets the query for an objective bank.

Multiple retrievals produce a nested `OR` term.

Returns the objective bank query

Return type `osid.learning.ObjectiveBankQuery`

Raise `Unimplemented` – `supports_objective_bank_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_objective_bank_query()‘‘ is ‘‘true‘‘.

`objective_bank_terms`

get_objective_query_record(*objective_record_type*)
Gets the objective query record corresponding to the given `Objective` record `Type`.

Multiple retrievals produce a nested `OR` term.

Parameters **objective_record_type** (`osid.type.Type`) – an objective query record type

Returns the objective query record

Return type `osid.learning.records.ObjectiveQueryRecord`

Raise `NullArgument` – `objective_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(objective_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Activity Query

class `dlkit.learning.queries.`**ActivityQuery**
Bases: *dlkit.osid.queries.OsidObjectQuery*, *dlkit.osid.queries.OsidSubjugateableQuery*

This is the query for searching activities.

Each method match request produces an `AND` term while multiple invocations of a method produces a nested `OR`.

match_objective_id(*objective_id*, *match*)
Sets the objective `Id` for this query.

Parameters

- **objective_id** (`osid.id.Id`) – an objective `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `objective_id` is `null`

compliance: mandatory – This method must be implemented.

objective_id_terms

supports_objective_query()
Tests if an `ObjectiveQuery` is available for querying objectives.

Returns `true` if an objective query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

objective_query
Gets the query for an objective.

Multiple retrievals produce a nested `OR` term.

Returns the objective query

Return type `osid.learning.ObjectiveQuery`

Raise `Unimplemented` – `supports_objective_query()` is `false`

compliance: optional – This method must be implemented if ``supports_objective_query()`` is ``true``.

`objective_terms`

`match_asset_id`(*asset_id*, *match*)

Sets the asset `Id` for this query.

Parameters

- **`asset_id`** (`osid.id.Id`) – an asset `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `asset_id` is `null`

compliance: mandatory – This method must be implemented.

`asset_id_terms`

`supports_asset_query`()

Tests if an `AssetQuery` is available for querying objectives.

Returns `true` if an robjective query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`asset_query`

Gets the query for an asset.

Multiple retrievals produce a nested `OR` term.

Returns the asset query

Return type `osid.repository.AssetQuery`

Raise `Unimplemented` – `supports_asset_query()` is `false`

compliance: optional – This method must be implemented if ``supports_objective_query()`` is ``true``.

`match_any_asset`(*match*)

Matches an activity that has any objective assigned.

Parameters **`match`** (`boolean`) – `true` to match activities with any asset, `false` to match activities with no asset

compliance: mandatory – This method must be implemented.

`asset_terms`

`match_course_id`(*course_id*, *match*)

Sets the course `Id` for this query.

Parameters

- **`course_id`** (`osid.id.Id`) – a course `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `course_id` is `null`

compliance: mandatory – This method must be implemented.

`course_id_terms`

`supports_course_query`()

Tests if a `CourseQuery` is available for querying courses.

Returns `true` if a course query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`course_query`
Gets the query for a course.

Multiple retrievals produce a nested `OR` term.

Returns the course query

Return type `osid.course.CourseQuery`

Raise `Unimplemented` – `supports_course_query()` is `false`

compliance: optional – This method must be implemented if ``supports_course_query()`` is ``true``.

`match_any_course`(*match*)
Matches an activity that has any course assigned.

Parameters **`match`** (`boolean`) – `true` to match activities with any courses, `false` to match activities with no courses

compliance: mandatory – This method must be implemented.

`course_terms`

`match_assessment_id`(*assessment_id*, *match*)
Sets the assessment `Id` for this query.

Parameters

- **`assessment_id`** (`osid.id.Id`) – an assessment `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `assessment_id` is `null`

compliance: mandatory – This method must be implemented.

`assessment_id_terms`

`supports_assessment_query`()
Tests if an `AssessmentQuery` is available for querying assessments.

Returns `true` if an assessment query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`assessment_query`
Gets the query for a assessment.

Multiple retrievals produce a nested `OR` term.

Returns the assessment query

Return type `osid.assessment.AssessmentQuery`

Raise `Unimplemented` – `supports_assessment_query()` is `false`

compliance: optional – This method must be implemented if ``supports_assessment_query()`` is ``true``.

`match_any_assessment`(*match*)
Matches an activity that has any assessment assigned.

Parameters **`match`** (`boolean`) – `true` to match activities with any assessments, `false` to match activities with no assessments

compliance: mandatory – This method must be implemented.

assessment_terms

match_objective_bank_id(*objective_bank_id*, *match*)
Sets the objective bank `Id` for this query.

Parameters

- **objective_bank_id** (`osid.id.Id`) – an objective bank `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `objective_bank_id` is `null`

compliance: mandatory – This method must be implemented.

objective_bank_id_terms

supports_objective_bank_query()
Tests if a `ObjectiveBankQuery` is available for querying resources.

Returns `true` if an objective bank query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

objective_bank_query
Gets the query for an objective bank.

Multiple retrievals produce a nested `OR` term.

Returns the objective bank query

Return type `osid.learning.ObjectiveBankQuery`

Raise `Unimplemented` – `supports_objective_bank_query()` is `false`

compliance: optional – This method must be implemented if ``supports_objective_bank_query()`` is ``true``.

objective_bank_terms

get_activity_query_record(*activity_record_type*)
Gets the activity query record corresponding to the given `Activity` record `Type`.

Multiple retrievals produce a nested `OR` term.

Parameters **activity_record_type** (`osid.type.Type`) – an activity query record type

Returns the activity query record

Return type `osid.learning.records.ActivityQueryRecord`

Raise `NullArgument` – `activity_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(activity_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Proficiency Query

class `dlkit.learning.queries.`**`ProficiencyQuery`**

Bases: *`dlkit.osid.queries.OsidRelationshipQuery`*

This is the query for searching proficiencies.

Each method match specifies an `AND` term while multiple invocations of the same method produce a nested `OR`.

`match_resource_id`(*resource_id*, *match*)

Sets the resource `Id` for this query.

Parameters

- **`resource_id`** (`osid.id.Id`) – a resource `Id`
- **`match`** (`boolean`) – `true` if a positive match, `false` for a negative match

Raise `NullArgument` – `resource_id` is `null`

compliance: mandatory – This method must be implemented.

`resource_id_terms`

`supports_resource_query`()

Tests if a `ResourceQuery` is available.

Returns `true` if a resource query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`resource_query`

Gets the query for a resource.

Multiple retrievals produce a nested `OR` term.

Returns the resource query

Return type `osid.resource.ResourceQuery`

Raise `Unimplemented` – `supports_resource_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_resource_query()‘‘ is ‘‘true‘‘.

`resource_terms`

`match_objective_id`(*objective_id*, *match*)

Sets the objective `Id` for this query.

Parameters

- **`objective_id`** (`osid.id.Id`) – an objective `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `objective_id` is `null`

compliance: mandatory – This method must be implemented.

`objective_id_terms`

`supports_objective_query`()

Tests if an `ObjectiveQuery` is available for querying objectives.

Returns `true` if an robjective query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

objective_query
Gets the query for an objective.

Multiple retrievals produce a nested `OR` term.

Returns the objective query

Return type `osid.learning.ObjectiveQuery`

Raise `Unimplemented` – `supports_objective_query()` is `false`

compliance: optional – This method must be implemented if ``supports_objective_query()`` is ``true``.

match_any_objective(*match*)
Matches an activity that has any objective assigned.

Parameters match (`boolean`) – `true` to match activities with any objective, `false` to match activities with no objective

compliance: mandatory – This method must be implemented.

objective_terms

match_completion(*start*, *end*, *match*)
Sets the completion for this query to match completion percentages between the given range inclusive.

Parameters

- **start** (`decimal`) – start of range
- **end** (`decimal`) – end of range
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `end` is less than `start`

compliance: mandatory – This method must be implemented.

completion_terms

match_minimum_completion(*completion*, *match*)
Sets the minimum completion for this query.

Parameters

- **completion** (`decimal`) – completion percentage
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

compliance: mandatory – This method must be implemented.

minimum_completion_terms

match_level_id(*grade_id*, *match*)
Sets the level grade `Id` for this query.

Parameters

- **grade_id** (`osid.id.Id`) – a grade `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `grade_id` is `null`

compliance: mandatory – This method must be implemented.

level_id_terms

supports_level_query()
Tests if a `GradeQuery` is available.

Returns `true` if a grade query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

level_query
Gets the query for a grade.

Multiple retrievals produce a nested `OR` term.

Returns the grade query

Return type `osid.grading.GradeQuery`

Raise `Unimplemented` – `supports_level_query()` is `false`

compliance: optional – This method must be implemented if ``supports_level_query()`` is ``true``.

match_any_level(*match*)
Matches an assessment offered that has any level assigned.

Parameters **match** (`boolean`) – `true` to match offerings with any level, `false` to match offerings with no levsls

compliance: mandatory – This method must be implemented.

level_terms

match_objective_bank_id(*objective_bank_id*, *match*)
Sets the objective bank `Id` for this query.

Parameters

- **objective_bank_id** (`osid.id.Id`) – an objective bank `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `objective_bank_id` is `null`

compliance: mandatory – This method must be implemented.

objective_bank_id_terms

supports_objective_bank_query()
Tests if a `ObjectiveBankQuery` is available for querying resources.

Returns `true` if an objective bank query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

objective_bank_query
Gets the query for an objective bank.

Multiple retrievals produce a nested `OR` term.

Returns the objective bank query

Return type `osid.learning.ObjectiveBankQuery`

Raise `Unimplemented` – `supports_objective_bank_query()` is `false`

compliance: optional – This method must be implemented if ``supports_objective_bank_query()`` is ``true``.

objective_bank_terms

get_proficiency_query_record(*proficiency_record_type*)
Gets the proficiency query record corresponding to the given `Proficiency` record `Type`.

Multiple retrievals produce a nested `OR` term.

Parameters **proficiency_record_type** (`osid.type.Type`) – a proficiency offered record type

Returns the proficiency offered query record

Return type `osid.learning.records.ProficiencyQueryRecord`

Raise `NullArgument` – `proficiency_offered_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(proficiency_offered_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Objective Bank Query

class `dlkit.learning.queries.`**ObjectiveBankQuery**
Bases: *dlkit.osid.queries.OsidCatalogQuery*

This is the query for searching objective banks.

Each method specifies an `AND` term while multiple invocations of the same method produce a nested `OR`.

match_objective_id(*objective_id*, *match*)
Sets the objective `Id` for this query.

Parameters

- **objective_id** (`osid.id.Id`) – an objective `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `objective_id` is `null`

compliance: mandatory – This method must be implemented.

objective_id_terms

supports_objective_query()
Tests if an `ObjectiveQuery` is available.

Returns `true` if an objective query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

objective_query
Gets the query for an objective.

Multiple retrievals produce a nested `OR` term.

Returns the objective query

Return type `osid.learning.ObjectiveQuery`

Raise `Unimplemented` – `supports_objective_query()` is `false` *compliance: optional – This method must be implemented if ‘‘supports_objective_query()‘‘ is ‘‘true‘‘.*

match_any_objective(*match*)
Matches an objective bank that has any objective assigned.

Parameters **match** (boolean) – true to match objective banks with any objective, false to match objective banks with no objectives

compliance: mandatory – This method must be implemented.

objective_terms

match_activity_id(*activity_id*, *match*)
Sets the activity Id for this query.

Parameters

- **activity_id** (osid.id.Id) – an activity Id
- **match** (boolean) – true for a positive match, false for a negative match

Raise NullArgument – activity_id is null

compliance: mandatory – This method must be implemented.

activity_id_terms

supports_activity_query()
Tests if a ActivityQuery is available for querying activities.

Returns true if an activity query is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

activity_query
Gets the query for an activity.

Multiple retrievals produce a nested OR term.

Returns the activity query

Return type osid.learning.ActivityQuery

Raise Unimplemented – supports_activity_query() is false

compliance: optional – This method must be implemented if ‘‘supports_activity_query()‘‘ is ‘‘true‘‘.

match_any_activity(*match*)
Matches an objective bank that has any activity assigned.

Parameters **match** (boolean) – true to match objective banks with any activity, false to match objective banks with no activities

compliance: mandatory – This method must be implemented.

activity_terms

match_ancestor_objective_bank_id(*objective_bank_id*, *match*)
Sets the objective bank Id for this query to match objective banks that have the specified objective bank as an ancestor.

Parameters

- **objective_bank_id** (osid.id.Id) – an objective bank Id
- **match** (boolean) – true for a positive match, false for a negative match

Raise `NullArgument` – `objective_bank_id` is `null`

compliance: mandatory – This method must be implemented.

`ancestor_objective_bank_id_terms`

`supports_ancestor_objective_bank_query``()`

Tests if a `ObjectiveBankQuery` is available for querying ancestor objective banks.

Returns `true` if an objective bank query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ancestor_objective_bank_query`

Gets the query for an objective bank.

Multiple retrievals produce a nested `OR` term.

Returns the objective bank query

Return type `osid.learning.ObjectiveBankQuery`

Raise `Unimplemented` – `supports_ancestor_objective_bank_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_ancestor_calndar_query()‘‘ is ‘‘true‘‘.

`match_any_ancestor_objective_bank``(`*match*`)`

Matches an objective bank that has any ancestor.

Parameters **`match`** (`boolean`) – `true` to match objective banks with any ancestor, `false` to match root objective banks

compliance: mandatory – This method must be implemented.

`ancestor_objective_bank_terms`

`match_descendant_objective_bank_id``(`*objective_bank_id*, *match*`)`

Sets the objective bank `Id` for this query to match objective banks that have the specified objective bank as a descendant.

Parameters

- **`objective_bank_id`** (`osid.id.Id`) – an objective bank `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `objective_bank_id` is `null`

compliance: mandatory – This method must be implemented.

`descendant_objective_bank_id_terms`

`supports_descendant_objective_bank_query``()`

Tests if a `ObjectiveBankQuery` is available for querying descendant objective banks.

Returns `true` if an objective bank query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`descendant_objective_bank_query`

Gets the query for an objective bank.

Multiple retrievals produce a nested `OR` term.

Returns the objective bank query

Return type `osid.learning.ObjectiveBankQuery`

Raise `Unimplemented` – `supports_descendant_objective_bank_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_descendant_calndar_query()‘‘ is ‘‘true‘‘.

`match_any_descendant_objective_bank`(*match*)
Matches an objective bank that has any descendant.

Parameters **`match`** (`boolean`) – `true` to match objective banks with any descendant, `false` to match leaf objective banks

compliance: mandatory – This method must be implemented.

`descendant_objective_bank_terms`

`get_objective_bank_query_record`(*objective_bank_record_type*)
Gets the objective bank query record corresponding to the given `ObjectiveBank` record `Type`.

Multiple record retrievals produce a nested `OR` term.

Parameters **`objective_bank_record_type`** (`osid.type.Type`) – an objective bank record type

Returns the objective bank query record

Return type `osid.learning.records.ObjectiveBankQueryRecord`

Raise `NullArgument` – `objective_bank_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(objective_bank_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Records

Objective Record

class `dlkit.learning.records.`**`ObjectiveRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `Objective`.

The methods specified by the record type are available through the underlying object.

Objective Query Record

class `dlkit.learning.records.`**`ObjectiveQueryRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `ObjectiveQuery`.

The methods specified by the record type are available through the underlying object.

Objective Form Record

class `dlkit.learning.records.`**`ObjectiveFormRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `ObjectiveForm`.

The methods specified by the record type are available through the underlying object.

Objective Search Record

class `dlkit.learning.records.`**`ObjectiveSearchRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `ObjectiveSearch`.

The methods specified by the record type are available through the underlying object.

Activity Record

class `dlkit.learning.records.`**`ActivityRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `Activity`.

The methods specified by the record type are available through the underlying object.

Activity Query Record

class `dlkit.learning.records.`**`ActivityQueryRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `ActivityQuery`.

The methods specified by the record type are available through the underlying object.

Activity Form Record

class `dlkit.learning.records.`**`ActivityFormRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `ActivityForm`.

The methods specified by the record type are available through the underlying object.

Activity Search Record

class `dlkit.learning.records.`**`ActivitySearchRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `ActivitySearch`.

The methods specified by the record type are available through the underlying object.

Proficiency Record

class `dlkit.learning.records.`**`ProficiencyRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `Proficiency`.

The methods specified by the record type are available through the underlying object.

Proficiency Query Record

class `dlkit.learning.records.`**`ProficiencyQueryRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `ProficiencyQuery`.

The methods specified by the record type are available through the underlying object.

Proficiency Form Record

class `dlkit.learning.records.`**`ProficiencyFormRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `ProficiencyForm`.

The methods specified by the record type are available through the underlying object.

Proficiency Search Record

class `dlkit.learning.records.`**`ProficiencySearchRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `ProficiencySearch`.

The methods specified by the record type are available through the underlying object.

Objective Bank Record

class `dlkit.learning.records.`**`ObjectiveBankRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `ObjectiveBank`.

The methods specified by the record type are available through the underlying object.

Objective Bank Query Record

class `dlkit.learning.records.`**`ObjectiveBankQueryRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `ObjectiveBankQuery`.

The methods specified by the record type are available through the underlying object.

Objective Bank Form Record

class `dlkit.learning.records.`**`ObjectiveBankFormRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `ObjectiveBankForm`.

The methods specified by the record type are available through the underlying object.

Objective Bank Search Record

class `dlkit.learning.records.`**`ObjectiveBankSearchRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `ObjectiveBankSearch`.

The methods specified by the record type are available through the underlying object.

Osid

Summary

Core Service Interface Definitions osid version 3.0.0

The Open Service Interface Definitions (OSIDs) is a service-based architecture to promote software interoperability. The OSIDs are a large suite of interface contract specifications that describe the integration points among services and system components for the purpose of creating choice among a variety of different and independently developed applications and systems, allowing independent evolution of software components within a complex system, and federated service providers.

The OSIDs were initially developed in 2001 as part of the MIT Open Knowledge Initiative Project funded by the Andrew W. Mellon Foundation to provide an architecture for higher education learning systems. OSID 3K development began in 2006 to redesign the capabilities of the specifications to apply to a much broader range of service domains and integration challenges among both small and large-scale enterprise systems.

The `osid` package defines the building blocks for the OSIDs which are defined in packages for their respective services. This package defines the top-level interfaces used by all the OSIDs as well as specification metadata and the OSID Runtime interface.

Meta Interfaces and Enumerations

- `OSID:` an enumeration listing the OSIDs defined in the specification.
- `Syntax:` an enumeration listing primitive types
- `Metadata:` an interface for describing data constraints on a data element

Interface Behavioral Markers

Interface behavioral markers are used to tag a behavioral pattern of the interface used to construct other object interfaces.

- `OsidPrimitive:` marks an OSID interface used as a primitive. OSID primitives may take the form interfaces if not bound to a language primitive. Interfaces used as primitives are marked to indicate that the underlying objects may be constructed by an OSID Consumer and an OSID Provider must honor any OSID primitive regardless of its origin.
- `Identifiable:` Marks an interface identifiable by an OSID `Id.`

- `Extensible:` Marks an interface as extensible through `OsidRecords.`
- `Browsable:` Marks an interface as providing `Property` inspection for its `OsidRecords.`
- `Suppliable:` Marks an interface as accepting data from an OSID Consumer.
- `Temporal:` Marks an interface that has a lifetime with begin an end dates.
- `Subjugateable:` Mars an interface that is dependent on another object.
- `Aggregateable:` Marks an interface that contains other objects normally related through other services.
- `Containable:` Marks an interface that contains a recursive reference to itself.
- `Sourceable:` Marks an interface as having a provider.
- `Federateable:` Marks an interface that can be federated using the OSID Hierarchy pattern.
- `Operable:` Marks an interface as responsible for performing operatons or tasks. `Operables` may be enabled or disabled.

Abstract service Interfaces

- `OsidProfile:` Defines interoperability methods used by OsidManagers.
- `OsidManager:` The entry point into an OSID and provides access to `OsidSessions.`
- `OsidProxyManager:` Another entry point into an OSID providing a means for proxying data from a middle tier application server to an underlying OSID Provider.
- `OsidSession` : A service interface accessible from an `OsidManager` that defines a set of methods for an aspect of a service.

Object-like interfaces are generally defined along lines of interoperability separating issues of data access from data management and searching. These interfaces may also implement any of the abstract behavioral interfaces listed above. The OSIDs do not adhere to a DAO/DTO model in its service definitions in that there are service methods defined on the objects (although they can be implemented using DTOs if desired). For the sake of an outline, we'll pretend they are data objects.

- `OsidObject:` Defines object data. `OsidObjects` are accessed from `OsidSessions. OsidObjects` are part of an interface hierarchy whose interfaces include the behavioral markers and a variety of common `OsidObjects.` All `OsidObjects` are `Identifiable, Extensible,` and have a `Type.` There are several variants of `OsidObjects` that indicate a more precise behavior.
- `OsidObjectQuery:` Defines a set of methods to query an OSID for its `OsidObjects` . An `OsidQuery` is accessed from an `OsidSession.`
- `OsidObjectQueryInspector:` Defines a set of methods to examine an `OsidQuery.`
- `OsidObjectForm:` Defines a set of methods to create and update data. `OsidForms` are accessed from `OsidSessions.`
- `OsidObjectSearchOrder:` Defines a set of methods to order search results. `OsidSearchOrders` are accessed from `OsidSessions.`

Most objects are or are derived from `OsidObjects`. Some object interfaces may not implement `OsidObejct` but instead derive directly from interface behavioral markers. Other `OsidObjects` may include interface behavioral markers to indicate functionality beyond a plain object. Several categories of `OsidObjects` have been defined to cluster behaviors to semantically distinguish their function in the OSIDs.

- `OsidCatalog:` At the basic level, a catalog represents a collection of other `OsidObjects.` The collection may be physical or virtual and may be federated to build larger `OsidCatalogs` using hierarchy services. `OsidCatalogs` may serve as a control point to filter or constrain the `OsidObjects` that may be visible or created. Each `OsidCatalog` may have its own provider identifty apart from the service provider.

- `OsidRelationship`: Relates two `OsidObjects`. The `OsidRelationship` represents the edge in a graph that may have its own relationship type and data. `OsidRelationships` are `Temporal` in that they have a time in which the relationship came into being and a time when the relationship ends.
- `OsidRule`: Defines an injection point for logic. An `OsidRule` may represent some constraint, evaluation, or execution. While authoring of `OsidRules` is outside the scope of the OSIDs, an `OsidRule` provides the mean to identify the rule and map it to certain `OsidObjects` to effect behavior of a service.

The most basic operations of an OSID center on retrieval, search, create & update, and notifications on changes to an `OsidObject`. The more advanced OSIDs model a system behavior where a variety of implicit relationships, constraints and rules come into play.

- `OsidGovernator`: Implies an activity or operation exists in the OSID Provider acting as an `Operable` point for a set of rules governing related `OsidObjects`. The `OsidGovernator` represents an engine of sorts in an OSID Provider and may have its own provider identity.
- `OsidCompendium` : `OsidObjects` which are reports or summaries based on transactional data managed elsewhere.

Managing data governing rules occurs in a separate set of interfaces from the effected `OsidObjects` (and often in a separate package). This allows for a normalized set of rules managing a small set of control points in a potentially large service.

- `OsidEnabler`: A managed control point to enable or disable the operation or effectiveness of another `OsidObject` . Enablers create a dynamic environment where behaviors and relationships can come and go based on rule evauations.
- `OsidConstrainer`: A managed control point to configure the constraints on the behavior of another `OsidObject`.
- `OsidProcessor`: A managed control point to configure the behavior of another `OsidObject` where some kins of processing is implied.

Other Abstract Interfaces

- `OsidSearch`: Defines set of methods to manage search options for performing searches.
- `OsidSearchResults`: Defines a set of methods to examine search results.
- `OsidReceiver`: Defines a set of methods invoked for asynchronous notification.
- `OsidList`: Defines a set of methods to sequentially access a set of objects.
- `OsidNode`: An interface used by hierarchy nodes.
- `OsidCondition`: An input or "statement of fact" into an `OsidRule` evaluation.
- `OsidInput`: An input of source data into an `OsidRule` processor.
- `OsidResult`: The output from processing an `OsidRule`.
- `OsidRecord`: An interface marker for an extension to another interface. `OsidRecord` are negotiated using OSID `Types`.
- `Property`: Maps a name to a value. Properties are available in OSID objects to provide a simplified view of data that may exist within a typed interface.
- `PropertyList`: A list of properties.

Runtime

- `OsidRuntimeProfile`: The `OsidProfile` for the runtime `OsidManager`.
- `OsidRuntimeManager`: The OSID Runtime service.

Abstract Flow

Generally, these definitions are abstract and not accesed directly. They are used as building blocks to define interfaces in the OSIDs themselves. OSIDs derive most of their definitions from a definition in the osid package. The methods that are defined at this abstract level versus the methods defined directly in a specific OSID is determined by the typing in the method signatures. The osid package interfaces are a means of ensuring consistency of common methods and not designed to facilitate object polymorphism among different OSIDs. A language binder may elect to alter the interface hierarchy presented in this specification and a provider need not parallel these interfaces in their implementations.

The flow of control through any OSID can be described in terms of these definitions. An `OsidManager` or `OsidProxyManager` is retrieved from the `OsidRuntimeManager` for a given service. Both types of managers share an interface for describing what they support in the `OsidProfile`.

`OsidSessions` are created from the `OsidManager`. `OsidSessions` tend to be organized along clusters of like-functionality. Lookup- oriented sessions retrieve `OsidObjects`. Return of multiple `OsidObjects` is done via the `OsidList`. Search-oriented sessions retrieve `OsidObjects` through searches provided through the `OsidQuery` and `OsidSearch` interfaces.

Administrative-oriented sessions create and update `OsidObjects` using the `OsidForm` interface. The `OsidForm` makes available `Metadata` to help define its rules for setting and changing various data elements.

`OsidObjects` can be organized within `OsidCatalogs`. An `OsidCatalog` is hierarchical and can be traversed through an `OsidNode`. An `OsidQuery` or an `OsidSearchOrder` may be mapped to a dynamic `OsidCatalog`. Such a query may be examined using an `OsidQueryInspector`.

A notification session provides a means for subscribing to events, “a new object has been created”, for example, and these events are received from an `OsidReceiver`.

Meta OSID Specification

The OSID Specification framework defines the interace and method structures as well as the language primitives and errors used throughout the OSIDs. The OSID Specifications are defined completely in terms of interfaces and the elements specified in the meta specification.

Language Primitives

Ths meta OSID Specification enumerates the allowable language primitives that can be used in OSID method signatures. Parameters and returns in OSID methods may be specified in terms of other OSID interfaces or using one of these primitives. An OSID Binder translates these language primitives into an appropriate language primitive counterpart.

An OSID Primitive differs from a language primitive. An OSID Primitive is an interface used to describe a more complex structure than a simple language primitive can support. Both OSID Primitives and language primitives have the same behavior in the OSIDs in that an there is no service encapsulation present allowing OSID Primitives to be consructed by an OSID Consumer.

Errors

OSID methods are required to return a value, if specified, or return one of the errors specified in the method signature. The meta package defines the set of errors that a method signtaure may use.

Errors should result when the contract of the interface as been violated or cannot be fulfilled and it is necessary to disrupt the flow of control for a consumer. Different errors are specified where it is forseen that a consumer may wish to execute a different action without violating the encapsulation of internal provider operations. Such actions do not include debugging or other detailed information which is the responsibility of the provider to manage. As such, the number of errors defined across all the interfaces is kept to a minimum and the context of the error may vary from method to method in accordance with the spceification.

Errors are categorized to convey the audience to which the error pertains.

- User Errors: Errors which may be the result of a user operation intended for the user.
- Operational Errors: Errors which may be the result of a system or some other problem intended for the user.

- Consumer Contract Errors: Software errors resulting in the use of the OSIDs by an OSID Consumer intended for the application programmer. These also include integration problems where the OSID Consumer bypassed a method to test for support of a service or type.
- Provider Contract Errors: Software errors in the use of an OSID by an OSID Provider intended for an implementation programmer.

Compliance

OSID methods include a compliance statement indicating whether a method is required or optional to implement. An optional OSID method is one that defines an UNIMPLEMENTED error and there is a corresponding method to test for the existence of an implementation.

OSID 3K Acknowledgements

- Tom Coppeto (Editor & Architect)
- Scott Thorne (Architect)

The authors gratefully acknowledge the following individuals for their time, wisdom, and contributions in shaping these specifications.

- Adam Franco, Middlebury College
- Jeffrey Merriman, Massachusetts Institute of Technology
- Charles Shubert, Massachusetts Insitute of Technology
- Prof. Marc Alier, Universitat Politècnica de Catalyuna
- Joshua Aresty, Massachusetts Institute of Technology
- Fabrizio Cardinali, Giunti Labs
- Pablo Casado, Universitat Politècnica de Catalyuna
- Alex Chapin, Middlebury College
- Craig Counterman, Massachusetts Institute of Technology
- Francesc Santanach Delisau, Universitat Oberta de Catalyuna
- Prof. Llorenç Valverde Garcia, Universitat Oberta de Catalyuna
- Catherine Iannuzzo, Massachusetts Institute of Technology
- Jeffrey Kahn, Verbena Consulting
- Michael Korcynski, Tufts University
- Anoop Kumar, Tufts University
- Eva de Lera, Universitat Oberta de Catalyuna
- Roberto García Marrodán, Universitat Oberta de Catalyuna
- Andrew McKinney, Massachusetts Institute of Technology
- Scott Morris, Apple
- Mark Norton, Nolaria Consulting
- Mark O'Neill, Dartmouth College
- Prof. Charles Severance, University of Michigan
- Stuart Sim, Sun Microsystems/Common Need
- Colin Smythe, IMS Global Learning Consortium

- George Ward, California State University
- Peter Wilkins, Massachusetts Institute of Technology
- Norman Wright, Massachusetts Institute of Technology

O.K.I. Acknowledgements

OSID 3K is based on the O.K.I. OSIDs developed as part of the MIT Open Knowledge Initiative (O.K.I) project 2001-2004.

- Vijay Kumar, O.K.I. Principal Investigator, Massachusetts Insitute of Technology
- Jeffrey Merriman, O.K.I. Project Director, Massachusetts Insitute of Technology
- Scott Thorne, O.K.I. Chief Architect, Massachusetts Institute of Technology
- Charles Shubert, O.K.I. Architect, Massachusetts Institute of Technology
- Lois Brooks, Project Coordinator, Stanford University
- Mark Brown, O.K.I. Project Manager, Massachusetts Institute of Technology
- Bill Fitzgerald, O.K.I. Finance Manager, Massachusetts Institute of Technology
- Judson Harward, Educational Systems Architect, Massachusetts Institute of Technology
- Charles Kerns, Educational Systems Architect, Stanford University
- Jeffrey Kahn, O.K.I. Partner, Verbena Consulting
- Judith Leonard, O.K.I. Project Administrator, Massachusetts Institute of Technology
- Phil Long, O.K.I. Outreach Coordinator, Massachusetts Institute of Technology
- Cambridge University, O.K.I. Core Collaborator
- Dartmouth College, O.K.I. Core Collaborator
- Massachusetts Institute of Technology, O.K.I. Core Collaborator
- North Carolina State University, O.K.I. Core Collaborator
- Stanford University, O.K.I. Core Collaborator
- University of Michigan, O.K.I. Core Collaborator
- University of Pennsylvania, O.K.I. Core Collaborator
- University of Wisconsin, Madison, O.K.I. Core Collaborator

Core Service Interface Definitions osid version 3.0.0

The Open Service Interface Definitions (OSIDs) is a service-based architecture to promote software interoperability. The OSIDs are a large suite of interface contract specifications that describe the integration points among services and system components for the purpose of creating choice among a variety of different and independently developed applications and systems, allowing independent evolution of software components within a complex system, and federated service providers.

The OSIDs were initially developed in 2001 as part of the MIT Open Knowledge Initiative Project funded by the Andrew W. Mellon Foundation to provide an architecture for higher education learning systems. OSID 3K development began in 2006 to redesign the capabilities of the specifications to apply to a much broader range of service domains and integration challenges among both small and large-scale enterprise systems.

The `osid` package defines the building blocks for the OSIDs which are defined in packages for their respective services. This package defines the top-level interfaces used by all the OSIDs as well as specification metadata and the OSID Runtime interface.

Meta Interfaces and Enumerations

- `OSID`: an enumeration listing the OSIDs defined in the specification.
- `Syntax`: an enumeration listing primitive types
- `Metadata`: an interface for describing data constraints on a data element

Interface Behavioral Markers

Interface behavioral markers are used to tag a behavioral pattern of the interface used to construct other object interfaces.

- `OsidPrimitive`: marks an OSID interface used as a primitive. OSID primitives may take the form interfaces if not bound to a language primitive. Interfaces used as primitives are marked to indicate that the underlying objects may be constructed by an OSID Consumer and an OSID Provider must honor any OSID primitive regardless of its origin.
- `Identifiable`: Marks an interface identifiable by an OSID `Id`.
- `Extensible`: Marks an interface as extensible through `OsidRecords`.
- `Browsable`: Marks an interface as providing `Property` inspection for its `OsidRecords`.
- `Suppliable`: Marks an interface as accepting data from an OSID Consumer.
- `Temporal`: Marks an interface that has a lifetime with begin an end dates.
- `Subjugateable`: Mars an interface that is dependent on another object.
- `Aggregateable`: Marks an interface that contains other objects normally related through other services.
- `Containable`: Marks an interface that contains a recursive reference to itself.
- `Sourceable`: Marks an interface as having a provider.
- `Federateable`: Marks an interface that can be federated using the OSID Hierarchy pattern.
- `Operable`: Marks an interface as responsible for performing operatons or tasks. `Operables` may be enabled or disabled.

Abstract service Interfaces

- `OsidProfile`: Defines interoperability methods used by OsidManagers.
- `OsidManager`: The entry point into an OSID and provides access to `OsidSessions`.
- `OsidProxyManager`: Another entry point into an OSID providing a means for proxying data from a middle tier application server to an underlying OSID Provider.
- `OsidSession` : A service interface accessible from an `OsidManager` that defines a set of methods for an aspect of a service.

Object-like interfaces are generally defined along lines of interoperability separating issues of data access from data management and searching. These interfaces may also implement any of the abstract behavioral interfaces listed above. The OSIDs do not adhere to a DAO/DTO model in its service definitions in that there are service methods defined on the objects (although they can be implemented using DTOs if desired). For the sake of an outline, we'll pretend they are data objects.

- `OsidObject`: Defines object data. `OsidObjects` are accessed from `OsidSessions`. `OsidObjects` are part of an interface hierarchy whose interfaces include the behavioral markers and a variety of common `OsidObjects`. All `OsidObjects` are `Identifiable`, `Extensible`, and have a `Type`. There are several variants of `OsidObjects` that indicate a more precise behavior.
- `OsidObjectQuery`: Defines a set of methods to query an OSID for its `OsidObjects` . An `OsidQuery` is accessed from an `OsidSession`.

- `OsidObjectQueryInspector`: Defines a set of methods to examine an `OsidQuery`.
- `OsidObjectForm`: Defines a set of methods to create and update data. `OsidForms` are accessed from `OsidSessions`.
- `OsidObjectSearchOrder`: Defines a set of methods to order search results. `OsidSearchOrders` are accessed from `OsidSessions`.

Most objects are or are derived from `OsidObjects`. Some object interfaces may not implement `OsidObejct` but instead derive directly from interface behavioral markers. Other `OsidObjects` may include interface behavioral markers to indicate functionality beyond a plain object. Several categories of `OsidObjects` have been defined to cluster behaviors to semantically distinguish their function in the OSIDs.

- `OsidCatalog`: At the basic level, a catalog represents a collection of other `OsidObjects`. The collection may be physical or virtual and may be federated to build larger `OsidCatalogs` using hierarchy services. `OsidCatalogs` may serve as a control point to filter or constrain the `OsidObjects` that may be visible or created. Each `OsidCatalog` may have its own provider identifty apart from the service provider.
- `OsidRelationship`: Relates two `OsidObjects`. The `OsidRelationship` represents the edge in a graph that may have its own relationship type and data. `OsidRelationships` are `Temporal` in that they have a time in which the relationship came into being and a time when the relationship ends.
- `OsidRule`: Defines an injection point for logic. An `OsidRule` may represent some constraint, evaluation, or execution. While authoring of `OsidRules` is outside the scope of the OSIDs, an `OsidRule` provides the mean to identify the rule and map it to certain `OsidObjects` to effect behavior of a service.

The most basic operations of an OSID center on retrieval, search, create & update, and notifications on changes to an `OsidObject`. The more advanced OSIDs model a system behavior where a variety of implicit relationships, constraints and rules come into play.

- `OsidGovernator`: Implies an activity or operation exists in the OSID Provider acting as an `Operable` point for a set of rules governing related `OsidObjects`. The `OsidGovernator` represents an engine of sorts in an OSID Provider and may have its own provider identity.
- `OsidCompendium` : `OsidObjects` which are reports or summaries based on transactional data managed elsewhere.

Managing data governing rules occurs in a separate set of interfaces from the effected `OsidObjects` (and often in a separate package). This allows for a normalized set of rules managing a small set of control points in a potentially large service.

- `OsidEnabler`: A managed control point to enable or disable the operation or effectiveness of another `OsidObject` . Enablers create a dynamic environment where behaviors and relationships can come and go based on rule evauations.
- `OsidConstrainer`: A managed control point to configure the constraints on the behavior of another `OsidObject`.
- `OsidProcessor`: A managed control point to configure the behavior of another `OsidObject` where some kins of processing is implied.

Other Abstract Interfaces

- `OsidSearch`: Defines set of methods to manage search options for performing searches.
- `OsidSearchResults`: Defines a set of methods to examine search results.
- `OsidReceiver`: Defines a set of methods invoked for asynchronous notification.
- `OsidList`: Defines a set of methods to sequentially access a set of objects.
- `OsidNode`: An interface used by hierarchy nodes.
- `OsidCondition`: An input or "statement of fact" into an `OsidRule` evaluation.

- `OsidInput`: An input of source data into an `OsidRule` processor.
- `OsidResult`: The output from processing an `OsidRule`.
- `OsidRecord`: An interface marker for an extension to another interface. `OsidRecord` are negotiated using OSID `Types`.
- `Property`: Maps a name to a value. Properties are available in OSID objects to provide a simplified view of data that may exist within a typed interface.
- `PropertyList`: A list of properties.

Runtime

- `OsidRuntimeProfile`: The `OsidProfile` for the runtime `OsidManager`.
- `OsidRuntimeManager`: The OSID Runtime service.

Abstract Flow

Generally, these definitions are abstract and not accesed directly. They are used as building blocks to define interfaces in the OSIDs themselves. OSIDs derive most of their definitions from a definition in the osid package. The methods that are defined at this abstract level versus the methods defined directly in a specific OSID is determined by the typing in the method signatures. The osid package interfaces are a means of ensuring consistency of common methods and not designed to facilitate object polymorphism among different OSIDs. A language binder may elect to alter the interface hierarchy presented in this specification and a provider need not parallel these interfaces in their implementations.

The flow of control through any OSID can be described in terms of these definitions. An `OsidManager` or `OsidProxyManager` is retrieved from the `OsidRuntimeManager` for a given service. Both types of managers share an interface for describing what they support in the `OsidProfile`.

`OsidSessions` are created from the `OsidManager`. `OsidSessions` tend to be organized along clusters of like-functionality. Lookup- oriented sessions retrieve `OsidObjects`. Return of multiple `OsidObjects` is done via the `OsidList`. Search-oriented sessions retrieve `OsidObjects` through searches provided through the `OsidQuery` and `OsidSearch` interfaces.

Administrative-oriented sessions create and update `OsidObjects` using the `OsidForm` interface. The `OsidForm` makes available `Metadata` to help define its rules for setting and changing various data elements.

`OsidObjects` can be organized within `OsidCatalogs`. An `OsidCatalog` is hierarchical and can be traversed through an `OsidNode`. An `OsidQuery` or an `OsidSearchOrder` may be mapped to a dynamic `OsidCatalog`. Such a query may be examined using an `OsidQueryInspector`.

A notification session provides a means for subscribing to events, “a new object has been created”, for example, and these events are received from an `OsidReceiver`.

Meta OSID Specification

The OSID Specification framework defines the interace and method structures as well as the language primitives and errors used throughout the OSIDs. The OSID Specifications are defined completely in terms of interfaces and the elements specified in the meta specification.

Language Primitives

Ths meta OSID Specification enumerates the allowable language primitives that can be used in OSID method signatures. Parameters and returns in OSID methods may be specified in terms of other OSID interfaces or using one of these primitives. An OSID Binder translates these language primitives into an appropriate language primitive counterpart.

An OSID Primitive differs from a language primitive. An OSID Primitive is an interface used to describe a more complex structure than a simple language primitive can support. Both OSID Primitives and language primitives have the same behavior in the OSIDs in that an there is no service encapsulation present allowing OSID Primitives to be consructed by an OSID Consumer.

Errors

OSID methods are required to return a value, if specified, or return one of the errors specified in the method signature. The meta package defines the set of errors that a method signtaure may use.

Errors should result when the contract of the interface as been violated or cannot be fulfilled and it is necessary to disrupt the flow of control for a consumer. Different errors are specified where it is forseen that a consumer may wish to execute a different action without violating the encapsulation of internal provider operations. Such actions do not include debugging or other detailed information which is the responsibility of the provider to manage. As such, the number of errors defined across all the interfaces is kept to a minimum and the context of the error may vary from method to method in accordance with the spceification.

Errors are categorized to convey the audience to which the error pertains.

- User Errors: Errors which may be the result of a user operation intended for the user.
- Operational Errors: Errors which may be the result of a system or some other problem intended for the user.
- Consumer Contract Errors: Software errors resulting in the use of the OSIDs by an OSID Consumer intended for the application programmer. These also include integration problems where the OSID Consumer bypassed a method to test for support of a service or type.
- Provider Contract Errors: Software errors in the use of an OSID by an OSID Provider intended for an implementation programmer.

Compliance

OSID methods include a compliance statement indicating whether a method is required or optional to implement. An optional OSID method is one that defines an UNIMPLEMENTED error and there is a corresponding method to test for the existence of an implementation.

OSID 3K Acknowledgements

- Tom Coppeto (Editor & Architect)
- Scott Thorne (Architect)

The authors gratefully acknowledge the following individuals for their time, wisdom, and contributions in shaping these specifications.

- Adam Franco, Middlebury College
- Jeffrey Merriman, Massachusetts Institute of Technology
- Charles Shubert, Massachusetts Insitute of Technology
- Prof. Marc Alier, Universitat Politècnica de Catalyuna
- Joshua Aresty, Massachusetts Institute of Technology
- Fabrizio Cardinali, Giunti Labs
- Pablo Casado, Universitat Politècnica de Catalyuna
- Alex Chapin, Middlebury College
- Craig Counterman, Massachusetts Institute of Technology
- Francesc Santanach Delisau, Universitat Oberta de Catalyuna
- Prof. Llorenç Valverde Garcia, Universitat Oberta de Catalyuna
- Catherine Iannuzzo, Massachusetts Institute of Technology
- Jeffrey Kahn, Verbena Consulting
- Michael Korcynski, Tufts University
- Anoop Kumar, Tufts University

- Eva de Lera, Universitat Oberta de Catalyuna
- Roberto García Marrodán, Universitat Oberta de Catalyuna
- Andrew McKinney, Massachusetts Institute of Technology
- Scott Morris, Apple
- Mark Norton, Nolaria Consulting
- Mark O'Neill, Dartmouth College
- Prof. Charles Severance, University of Michigan
- Stuart Sim, Sun Microsystems/Common Need
- Colin Smythe, IMS Global Learning Consortium
- George Ward, California State University
- Peter Wilkins, Massachusetts Institute of Technology
- Norman Wright, Massachusetts Institute of Technology

O.K.I. Acknowledgements

OSID 3K is based on the O.K.I. OSIDs developed as part of the MIT Open Knowledge Initiative (O.K.I) project 2001-2004.

- Vijay Kumar, O.K.I. Principal Investigator, Massachusetts Insitute of Technology
- Jeffrey Merriman, O.K.I. Project Director, Massachusetts Insitute of Technology
- Scott Thorne, O.K.I. Chief Architect, Massachusetts Institute of Technology
- Charles Shubert, O.K.I. Architect, Massachusetts Institute of Technology
- Lois Brooks, Project Coordinator, Stanford University
- Mark Brown, O.K.I. Project Manager, Massachusetts Institute of Technology
- Bill Fitzgerald, O.K.I. Finance Manager, Massachusetts Institute of Technology
- Judson Harward, Educational Systems Architect, Massachusetts Institute of Technology
- Charles Kerns, Educational Systems Architect, Stanford University
- Jeffrey Kahn, O.K.I. Partner, Verbena Consulting
- Judith Leonard, O.K.I. Project Administrator, Massachusetts Institute of Technology
- Phil Long, O.K.I. Outreach Coordinator, Massachusetts Institute of Technology
- Cambridge University, O.K.I. Core Collaborator
- Dartmouth College, O.K.I. Core Collaborator
- Massachusetts Institute of Technology, O.K.I. Core Collaborator
- North Carolina State University, O.K.I. Core Collaborator
- Stanford University, O.K.I. Core Collaborator
- University of Michigan, O.K.I. Core Collaborator
- University of Pennsylvania, O.K.I. Core Collaborator
- University of Wisconsin, Madison, O.K.I. Core Collaborator

Service Managers

Osid Manager

class `dlkit.services.osid.`**`OsidManager`**

Bases: `dlkit.services.osid.OsidProfile`

The `OsidManager` is the top level interface for all OSID managers.

An OSID manager is instantiated through the `OsidRuntimeManager` and represents an instance of a service. An OSID manager is responsible for implementing a profile for a service and creating sessions that, in general, correspond to the profile. An application need only create a single `OsidManager` per service and implementors must ensure the `OsidManager` is thread-safe “`. The ``OsidSessions` spawned from an OSID manager are dedicated to single processing threads. The `OsidManager` defines methods in common throughout all OSID managers which implement this interface.

`initialize`(*runtime*)

Initializes this manager.

A manager is initialized once at the time of creation.

Parameters **`runtime`** (`osid.OsidRuntimeManager`) – the runtime environment

Raise `ConfigurationError` – an error with implementation configuration

Raise `IllegalState` – this manager has already been initialized by the `OsidRuntime`

Raise `NullArgument` – `runtime` is `null`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented. implementation notes: In addition to loading its runtime configuration an implementation may create shared resources such as connection pools to be shared among all sessions of this service and released when this manager is closed. Providers must thread-protect any data stored in the manager. To maximize interoperability, providers should not honor a second call to `initialize()` and must set an `IllegalState` error.

`rollback_service`(*rollback_time*)

Rolls back this service to a point in time.

Parameters **`rollback_time`** (`timestamp`) – the requested time

Returns the journal entry corresponding to the actual state of this service

Return type `osid.journaling.JournalEntry`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unimplemented` – `supports_journal_rollback()` is `false`

compliance: mandatory – This method must be implemented.

`change_branch`(*branch_id*)

Changes the service branch.

Parameters **`branch_id`** (`osid.id.Id`) – the new service branch

Raise `NotFound` – `branch_id` not found

Raise `NullArgument` – `branch_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unimplemented` – `supports_journal_branching()` is `false`

compliance: mandatory – This method must be implemented.

Osid Proxy Manager

class `dlkit.services.osid.`**`OsidProxyManager`**

Bases: `dlkit.services.osid.OsidProfile`

The `OsidProxyManager` is the top level interface for all OSID proxy managers.

A proxy manager accepts parameters to pass through end-user authentication credentials and other necessary request parameters in a server environment. Native applications should use an `OsidManager` to maintain a higher degree of interoperability by avoiding this coupling.

An OSID proxy manager is instantiated through the `OsidRuntimeManager` and represents an instance of a service. An OSID manager is responsible for defining clusters of interoperability within a service and creating sessions that generally correspond to these clusters, An application need only create a single `OsidProxyManager` per service and implementors must ensure the `OsidProxyManager` is thread-safe **“**`. The ``OsidSessions` spawned from an OSID manager are dedicated to single processing threads. The `OsidProxyManager` defines methods in common throughout all OSID managers which implement this interface.

`initialize`(*runtime*)

Initializes this manager.

A manager is initialized once at the time of creation.

Parameters **`runtime`** (`osid.OsidRuntimeManager`) – the runtime environment

Raise `ConfigurationError` – an error with implementation configuration

Raise `IllegalState` – this manager has already been initialized by the `OsidRuntime`

Raise `NullArgument` – `runtime` is `null`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented. implementation notes: In addition to loading its runtime configuration an implementation may create shared resources such as connection pools to be shared among all sessions of this service and released when this manager is closed. Providers must thread-protect any data stored in the manager. To maximize interoperability, providers should not honor a second call to `initialize()` and must set an `IllegalState` error.

`rollback_service`(*rollback_time*, *proxy*)

Rolls back this service to a point in time.

Parameters

- **`rollback_time`** (`timestamp`) – the requested time
- **`proxy`** (`osid.proxy.Proxy`) – a proxy

Returns the journal entry corresponding to the actual state of this service

Return type `osid.journaling.JournalEntry`

Raise `NullArgument` – `proxy` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unimplemented` – `supports_journal_rollback()` is `false`

compliance: mandatory – This method must be implemented.

change_branch (*branch_id*, *proxy*)

Changes the service branch.

Parameters

- **branch_id** (`osid.id.Id`) – the new service branch
- **proxy** (`osid.proxy.Proxy`) – a proxy

Raise `NotFound` – `branch_id` not found

Raise `NullArgument` – `branch_id` or `proxy` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unimplemented` – `supports_journal_branching()` is `false`

compliance: mandatory – This method must be implemented.

Osid Runtime Profile

class `dlkit.services.osid.`**OsidRuntimeProfile**

Bases: `dlkit.services.osid.OsidProfile`

The `OsidRuntimeProfile` defines the service aspects of the OSID runtime service.

supports_configuration ()

Tests if a configuration service is provided within this runtime environment.

Returns `true` if a configuration service is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

Osid Runtime Manager

class `dlkit.services.osid.`**OsidRuntimeManager**

Bases: *dlkit.services.osid.OsidManager*, *dlkit.services.osid.OsidRuntimeProfile*

The `OsidRuntimeManager` represents and OSID platform and contains the information required for running OSID implementations such as search paths and configurations.

get_manager (*osid*, *impl_class_name*, *version*)

Finds, loads and instantiates providers of OSID managers.

Providers must conform to an OsidManager interface. The interfaces are defined in the OSID enumeration. For all OSID requests, an instance of `OsidManager` that implements the `OsidManager` interface is returned. In bindings where permitted, this can be safely cast into the requested manager.

Parameters

- **osid** (`osid.OSID`) – represents the OSID
- **impl_class_name** (`string`) – the name of the implementation
- **version** (`osid.installation.Version`) – the minimum required OSID specification version

Returns the manager of the service

Return type `osid.OsidManager`

Raise `ConfigurationError` – an error in configuring the implementation

Raise `NotFound` – the implementation class was not found

Raise `NullArgument` – `impl_class_name` or `version` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `impl_class_name` does not support the requested OSID

compliance: mandatory – This method must be implemented. implementation notes: After finding and instantiating the requested `OsidManager,` providers must invoke `OsidManager.initialize(OsidRuntimeManager)` where the environment is an instance of the current environment that includes the configuration for the service being initialized. The `OsidRuntimeManager` passed may include information useful for the configuration such as the identity of the service being instantiated.

`get_proxy_manager`(*osid*, *implementation*, *version*)

Finds, loads and instantiates providers of OSID managers.

Providers must conform to an `OsidManager` interface. The interfaces are defined in the OSID enumeration. For all OSID requests, an instance of `OsidManager` that implements the `OsidManager` interface is returned. In bindings where permitted, this can be safely cast into the requested manager.

Parameters

- **`osid`** (`osid.OSID`) – represents the OSID
- **`implementation`** (`string`) – the name of the implementation
- **`version`** (`osid.installation.Version`) – the minimum required OSID specification version

Returns the manager of the service

Return type `osid.OsidProxyManager`

Raise `ConfigurationError` – an error in configuring the implementation

Raise `NotFound` – the implementation class was not found

Raise `NullArgument` – `implementation` or `version` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `implementation` does not support the requested OSID

compliance: mandatory – This method must be implemented. implementation notes: After finding and instantiating the requested `OsidManager,` providers must invoke `OsidManager.initialize(OsidRuntimeManager)` where the environment is an instance of the current environment that includes the configuration for the service being initialized. The `OsidRuntimeManager` passed may include information useful for the configuration such as the identity of the service being instantiated.

`configuration`

Gets the current configuration in the runtime environment.

Returns a configuration

Return type `osid.configuration.ValueLookupSession`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – an authorization failure occured

Raise `Unimplemented` – a configuration service is not supported

compliance: optional – This method must be implemented if ``supports_configuration()`` is ``true``.

Objects

Osid Object

class `dlkit.osid.objects.`**`OsidObject`**

Bases: *`dlkit.osid.markers.Identifiable`*, *`dlkit.osid.markers.Extensible`*, *`dlkit.osid.markers.Browsable`*

`OsidObject` is the top level interface for all OSID Objects.

An OSID Object is an object identified by an OSID `Id` and may implements optional interfaces. OSID Objects also contain a display name and a description. These fields are required but may be used for a variety of purposes ranging from a primary name and description of the object to a more user friendly display of various attributes.

Creation of OSID Objects and the modification of their data is managed through the associated `OsidSession` which removes the dependency of updating data elements upon object retrieval.The `OsidManager` should be used to test if updates are available and determine what `PropertyTypes` are supported. The `OsidManager` is also used to create the appropriate `OsidSession` for object creation, updates and deletes.

All `OsidObjects` are identified by an immutable `Id`. An `Id` is assigned to an object upon creation of the object and cannot be changed once assigned.

An `OsidObject` may support one or more supplementary records which are expressed in the form of interfaces. Each record interface is identified by a Type. A record interface may extend another record interface where support of the parent record interface is implied. In this case of interface inheritance, support of the parent record type may be implied through `has_record_type()` and not explicit in `getRecordTypes()`.

For example, if recordB extends recordA, typeB is a child of typeA. If a record implements typeB, than it also implements typeA. An application that only knows about typeA retrieves recordA. An application that knows about typeB, retrieves recordB which is the union of methods specified in typeA and typeB. If an application requests typeA, it may not attempt to access methods defined in typeB as they may not exist until explicitly requested. The mechanics of this polymorphism is defined by the language binder. One mechanism might be the use of casting.

In addition to the record `Types,` OSID Objects also have a genus `Type`. A genus `Type` indicates a classification or kind of the object where an "is a" relationship exists. The purpose of of the genus `Type` is to avoid the creation of unnecessary record types that may needlessly complicate an interface hierarchy or introduce interoperability issues. For example, an OSID object may have a record `Type` of `Publication` that defines methods pertinent to publications, such as an ISBN number. A provider may wish to distinguish between books and journals without having the need of new record interfaces. In this case, the genus `Type` may be one of `Book` or `Journal`. While this distinction can aid a search, these genres should be treated in such a way that do not introduce interoperability problems.

Like record Types, the genus Types may also exist in an implicit type hierarchy. An OSID object always has at least one genus. Genus types should not be confused with subject tagging, which is managed externally to the object. Unlike record `Types,` an object's genus may be modified. However, once an object's record is created with a record `Type,` it cannot be changed.

Methods that return values are not permitted to return nulls. If a value is not set, it is indicated in the `Metadata` of the update form.

display_name

Gets the preferred display name associated with this instance of this OSID object appropriate for display to the user.

Returns the display name

Return type `osid.locale.DisplayText`

compliance: mandatory – This method must be implemented. implementation notes: A display name is a string used for identifying an object in human terms. A provider may wish to initialize the display name based on one or more object attributes. In some cases, the display name may not map to a specific or significant object attribute but simply be used as a preferred display name that can be modified. A provider may also wish to translate the display name into a specific locale using the Locale service. Some OSIDs define methods for more detailed naming.

description

Gets the description associated with this instance of this OSID object.

Returns the description

Return type `osid.locale.DisplayText`

compliance: mandatory – This method must be implemented. implementation notes: A description is a string used for describing an object in human terms and may not have significance in the underlying system. A provider may wish to initialize the description based on one or more object attributes and/or treat it as an auxiliary piece of data that can be modified. A provider may also wish to translate the description into a specific locale using the Locale service.

genus_type

Gets the genus type of this object.

Returns the genus type of this object

Return type `osid.type.Type`

compliance: mandatory – This method must be implemented.

is_of_genus_type(*genus_type*)

Tests if this object is of the given genus `Type`.

The given genus type may be supported by the object through the type hierarchy.

Parameters **genus_type** (`osid.type.Type`) – a genus type

Returns `true` if this object is of the given genus `Type, false` otherwise

Return type `boolean`

Raise `NullArgument` – `genus_type` is `null`

compliance: mandatory – This method must be implemented.

Osid Relationship

class `dlkit.osid.objects.`**OsidRelationship**

Bases: *`dlkit.osid.objects.OsidObject`*, *`dlkit.osid.markers.Temporal`*

A `Relationship` associates two OSID objects.

Relationships are transient. They define a date range for which they are in effect.

Unlike other `OsidObjects` that rely on the auxiliary Journaling OSID to track variance over time, `OsidRelationships` introduce a different concept of time independent from journaling. For example, in the present, a student was registered in a course and dropped it. The relationship between the student and the course remains pertinent, independent of any journaled changes that may have occurred to either the student or the course.

Once the student has dropped the course, the relationship has expired such that `is_effective()` becomes false. It can be inferred that during the period of the effective dates, the student was actively registered in the course. Here is an example:

- T1. September 1: Student registers for course for grades
- T2. September 10: Student drops course
- T3. September 15: Student re-registers for course pass/fail

The relationships are: T1. R1 {effective, September 1 -> end of term, data=grades} T2. R1 {ineffective, September 1 -> September 10, data=grades} T3. R1 {ineffective, September 1 -> September 10, data=grades}

R2 {effective, September 10 -> end of term, data=p/f}

An OSID Provider may also permit dates to be set in the future in which case the relationship can become automatically become effective at a future time and later expire. More complex effectiveness management can be done through other rule-based services.

OSID Consumer lookups and queries of relationships need to consider that it may be only effective relationshps are of interest.

`has_end_reason``()`

Tests if a reason this relationship came to an end is known.

Returns `true` if an end reason is available, `false` otherwise

Return type `boolean`

Raise `IllegalState` – `is_effective()` is `true`

compliance: mandatory – This method must be implemented.

`end_reason_id`

Gets a state `Id` indicating why this relationship has ended.

Returns a state `Id`

Return type `osid.id.Id`

Raise `IllegalState` – `has_end_reason()` is `false`

compliance: mandatory – This method must be implemented.

`end_reason`

Gets a state indicating why this relationship has ended.

Returns a state

Return type `osid.process.State`

Raise `IllegalState` – `has_end_reason()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Osid Catalog

class `dlkit.osid.objects.`**`OsidCatalog`**

Bases: *`dlkit.osid.objects.OsidObject`*, *`dlkit.osid.markers.Sourceable`*, *`dlkit.osid.markers.Federateable`*

`OsidCatalog` is the top level interface for all OSID catalog-like objects.

A catalog relates to other OSID objects for the purpose of organization and federation and almost always are hierarchical. An example catalog is a `Repository` that relates to a collection of `Assets`.

`OsidCatalogs` allow for the retrieval of a provider identity and branding.

Collections visible through an `OsidCatalog` may be the output of a dynamic query or some other rules-based evaluation. The facts surrounding the evaluation are the `OsidObjects` visible to the `OsidCatalog` from its position in the federated hierarchy. The input conditions may satisifed on a service-wide basis using an `OsidQuery` or environmental conditions supplied to the services via a `Proxy` .

Often, the selection of an `OsidCatalog` in instantiating an `OsidSession` provides access to a set of `OsidObjects` . Because the view inside an `OsidCatalog` can also be produced behaviorally using a rules evaluation, the `Id` (or well-known alias) of the `OsidCatalog` may be used as an abstract means of requesting a predefined set of behaviors or data constraints from an OSID Provider.

The flexibility of interpretation together with its central role in federation to build a rich and complex service from a set of individual OSID Providers makes cataloging an essential pattern to achieve abstraction from implementations in the OSIDs without loss of functionality. Most OSIDs include a cataloging pattern.

Osid Rule

class `dlkit.osid.objects.`**`OsidRule`**

Bases: *`dlkit.osid.objects.OsidObject`*, *`dlkit.osid.markers.Operable`*

An `OsidRule` identifies an explicit or implicit rule evaluation.

An associated `Rule` may be available in cases where the behavior of the object can be explicitly modified using a defined rule. In many cases, an `OsidObject` may define specific methods to manage certain common behavioral aspects and delegate anything above and beyond what has been defined to a rule evaluation.

Rules are defined to be operable. In the case of a statement evaluation, an enabled rule overrides any evaluation to return `true` and a disabled rule overrides any evaluation to return `false`.

`Rules` are never required to consume or implement. They serve as a mechanism to offer a level of management not attainable in the immediate service definition. Each Rule implies evaluating a set of facts known to the service to produce a resulting beavior. Rule evaluations may also accept input data or conditions, however, `OsidRules` as they appear in throughout the services may or may not provide a means of supplying `OsidConditions` directly. In the services where an explicit `OsidCondition` is absent they may be masquerading as another interface such as a `Proxy` or an `OsidQuery` .

`has_rule``()`

Tests if an explicit rule is available.

Returns `true` if an explicit rule is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`rule_id`

Gets the explicit rule `Id`.

Returns the rule `Id`

Return type `osid.id.Id`

Raise `IllegalState` – `has_rule()` is `false`

compliance: mandatory – This method must be implemented.

rule

Gets the explicit rule.

Returns the rule

Return type `osid.rules.Rule`

Raise `IllegalState` – `has_rule()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Osid Enabler

class `dlkit.osid.objects.`**`OsidEnabler`**

Bases: *`dlkit.osid.objects.OsidRule`*, *`dlkit.osid.markers.Temporal`*

`OsidEnabler` is used to manage the effectiveness, enabledness, or operation of an `OsidObejct`.

The `OsidEnabler` itself is active or inactive When an `OsidEnabler` is active, any `OsidObject` mapped to it is "on." When all `OsidEnablers` mapped to an `OsidObject` are inactive, then the `OsidObject` is "off."

The managed `OsidObject` may have varying semantics as to what its on/off status means and in particular, which methods are used to indicate the effect of an `OsidEnabler`. Some axamples:

- `Operables: OsidEnablers` effect the operational status.
- `Temporals: OsidEnablers` may be used to extend or shorten the effectiveness of a `Temporal` such as an `OsidRelationship.`

In the case where an `OsidEnabler` may cause a discontinuity in a `Temporal,` the `OsidEnabler` may cause the creation of new `Temporals` to capture the gap in effectiveness.

For example, An `OsidRelationship` that began in 2007 may be brought to an end in 2008 due to the absence of any active `OsidEnablers.` When an effective `OsidEnabler` appears in 2009, a new `OsidRelationship` is created with a starting effective date of 2009 leaving the existing `OsidRelationship` with effective dates from 2007 to 2008.

An `OsidEnabler` itself is both a `Temporal` and an `OsidRule` whose activity status of the object may be controlled administratively, using a span of effective dates, through an external rule, or all three. The `OsidEnabler` defines a set of canned rules based on dates, events, and cyclic events.

is_effective_by_schedule()

Tests if the effectiveness of the enabler is governed by a `Schedule`.

If a schedule exists, it is bounded by the effective dates of this enabler. If `is_effective_by_schedule()` is `true,` `is_effective_by_event()` and `is_effective_by_cyclic_event()` must be `false`.

Returns `true` if the enabler is governed by schedule, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

schedule_id

Gets the schedule `Id`.

Returns the schedule `Id`

Return type `osid.id.Id`

Raise `IllegalState` – `is_effective_by_schedule()` is `false`

compliance: mandatory – This method must be implemented.

schedule

Gets the schedule.

Returns the schedule

Return type `osid.calendaring.Schedule`

Raise `IllegalState` – `is_effective_by_schedule()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

is_effective_by_event()

Tests if the effectiveness of the enabler is governed by an `Event` such that the start and end dates of the event govern the effectiveness.

The event may also be a `RecurringEvent` in which case the enabler is effective for start and end dates of each event in the series If an event exists, it is bounded by the effective dates of this enabler. If `is_effective_by_event()` is `true`, `is_effective_by_schedule()` and `is_effective_by_cyclic_event()` must be `false`.

Returns `true` if the enabler is governed by an event, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

event_id

Gets the event `Id`.

Returns the event `Id`

Return type `osid.id.Id`

Raise `IllegalState` – `is_effective_by_event()` is `false`

compliance: mandatory – This method must be implemented.

event

Gets the event.

Returns the event

Return type `osid.calendaring.Event`

Raise `IllegalState` – `is_effective_by_event()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

is_effective_by_cyclic_event()

Tests if the effectiveness of the enabler is governed by a `CyclicEvent`.

If a cyclic event exists, it is evaluated by the accompanying cyclic time period. If `is_effective_by_cyclic_event()` is `true, is_effective_by_schedule()` and `is_effective_by_event()` must be `false`.

Returns `true` if the enabler is governed by a cyclic event, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`cyclic_event_id`

Gets the cyclic event `Id`.

Returns the cyclic event `Id`

Return type `osid.id.Id`

Raise `IllegalState` – `is_effective_by_cyclic_event()` is `false`

compliance: mandatory – This method must be implemented.

`cyclic_event`

Gets the cyclic event.

Returns the cyclic event

Return type `osid.calendaring.cycle.CyclicEvent`

Raise `IllegalState` – `is_effective_by_cyclic_event()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`is_effective_for_demographic``()`

Tests if the effectiveness of the enabler applies to a demographic resource.

Returns `true` if the rule apples to a demographic. `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`demographic_id`

Gets the demographic resource `Id`.

Returns the resource `Id`

Return type `osid.id.Id`

Raise `IllegalState` – `is_effective_for_demographic()` is `false`

compliance: mandatory – This method must be implemented.

`demographic`

Gets the demographic resource.

Returns the resource representing the demographic

Return type `osid.resource.Resource`

Raise `IllegalState` – `is_effective_for_demographic()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Osid Constrainer

class `dlkit.osid.objects.`**`OsidConstrainer`**

Bases: *`dlkit.osid.objects.OsidRule`*

An `OsidConstrainer` marks an interface as a control point to constrain another object.

A constrainer may define specific methods to describe the constrainment or incorporate external logic using a rule.

Osid Processor

class `dlkit.osid.objects.`**`OsidProcessor`**

Bases: *`dlkit.osid.objects.OsidRule`*

An `OsidProcessor` is an interface describing the operation of another object.

A processor may define specific methods to manage processing, or incorporate external logic using a rule.

Osid Governator

class `dlkit.osid.objects.`**`OsidGovernator`**

Bases: *`dlkit.osid.objects.OsidObject`*, *`dlkit.osid.markers.Operable`*, *`dlkit.osid.markers.Sourceable`*

An `OsidGovernator` is a control point to govern the behavior of a service.

`OsidGovernators` generally indicate the presence of `OsidEnablers` and other rule governing interfaces to provide a means of managing service operations and constraints from a "behind the scenes" perspective. The `OsidGovernator` is a focal point for these various rules.

`OsidGovernators` are `Sourceable`. An `OsidGovernator` implies a governance that often corresponds to a provider of a process as opposed to a catalog provider of `OsidObjects`.

`OsidGovernators` are `Operable`. They indicate an active and operational status and related rules may be administratively overridden using this control point. Administratively setting the enabled or disabled flags in the operator overrides any enabling rule mapped to this `OsidGovernator`.

Osid Compendium

class `dlkit.osid.objects.`**`OsidCompendium`**

Bases: *`dlkit.osid.objects.OsidObject`*, *`dlkit.osid.markers.Subjugateable`*

`OsidCompendium` is the top level interface for reports based on measurements, calculations, summaries, or views of transactional activity within periods of time.

This time dimension of this report may align with managed time periods, specific dates, or both. Oh my.

Reports are often derived dynamically based on an examination of data managed elsewhere in an OSID. Reports may also be directly managed outside where it is desirable to capture summaries without the detail of the implied evaluated data. The behavior of a direct create or update of a report is not specified but is not limited to an override or a cascading update of underlying data.

The start and end date represents the date range used in the evaluation of the transactional data on which this report is based. The start and end date may be the same indicating that the evaluation occurred at a point in time rather than across a date range. The start and end date requested may differ from the start and end date indicated in this report because of the inability to interpolate or extrapolate the date. These dates should be examined to understand what actually occurred and to what dates the information in this report pertains.

These dates differ from the dates the report itself was requested, created, or modified. The dates refer to the context of the evaluation. In a managed report, the dates are simply the dates to which the report information pertains. The history of a single report may be examined in the Journaling OSID.

For example, the Location of a Resource at 12:11pm is reported to be in Longwood and at 12:23pm is reported to be at Chestnut Hill. A request of a `ResourceLocation`. A data correction may update the Longwood time to be 12:09pm. The update of the `ResourceLocation` from 12:11pm to 12:09pm may be examined in the Journaling OSID while the 12:11pm time would not longer be visible in current versions of this report.

Reports may be indexed by a managed time period such as a `Term` or `FiscalPeriod`. The evaluation dates may map to the opening and closing dates of the time period. Evaluation dates that differ from the time period may indicate that the transactional data is incomplete for that time period or that the report was calculated using a requested date range.

`OsidCompendiums` are subjugates to other `OsidObjects` in that what is reported is tied to an instance of a dimension such as a person, account, or an `OsidCatalog` .

`start_date`

Gets the start date used in the evaluation of the transactional data on which this report is based.

Returns the date

Return type `osid.calendaring.DateTime`

compliance: mandatory – This method must be implemented.

`end_date`

Gets the end date used in the evaluation of the transactional data on which this report is based.

Returns the date

Return type `osid.calendaring.DateTime`

compliance: mandatory – This method must be implemented.

`is_interpolated``()`

Tests if this report is interpolated within measured data or known transactions.

Interpolation may occur if the start or end date fall between two known facts or managed time period.

Returns `true` if this report is interpolated, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`is_extrapolated``()`

Tests if this report is extrapolated outside measured data or known transactions.

Extrapolation may occur if the start or end date fall outside two known facts or managed time period. Extrapolation may occur within a managed time period in progress where the results of the entire time period are projected.

Returns `true` if this report is extrapolated, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

Osid Capsule

class `dlkit.osid.objects.`**`OsidCapsule`**

`OsidCapsule` wraps other objects.

The interface has no meaning other than to return a set of semantically unrelated objects from a method.

Osid Form

class dlkit.osid.objects.**OsidForm**

Bases: *dlkit.osid.markers.Identifiable*, *dlkit.osid.markers.Suppliable*

The `OsidForm` is the vehicle used to create and update objects.

The form is a container for data to be sent to an update or create method of a session. Applications should persist their own data until a form is successfully submitted in an update or create transaction.

The form may provide some feedback as to the validity of certain data updates before the update transaction is issued to the correspodning session but a successful modification of the form is not a guarantee of success for the update transaction. A consumer may elect to perform all updates within a single update transaction or break up a large update intio smaller units. The tradeoff is the granularity of error feedback vs. the performance gain of a single transaction.

`OsidForms` are `Identifiable`. The `Id` of the `OsidForm` is used to uniquely identify the update or create transaction and not that of the object being updated. Currently, it is not necessary to have these `Ids` persisted.

As with all aspects of the OSIDs, nulls cannot be used. Methods to clear values are also defined in the form.

A new `OsidForm` should be acquired for each transaction upon an `OsidObject`. Forms should not be reused from one object to another even if the supplied data is the same as the forms may encapsulate data specific to the object requested. Example of changing a display name and a color defined in a color interface extension:

> ObjectForm form = session.getObjectFormForUpdate(objectId); form.setDisplayName("new name"); ColorForm recordForm = form.getFormRecord(colorRecordType); recordForm.setColor("green"); session.updateObject(objectId, form);

is_for_update()

Tests if this form is for an update operation.

Returns `true` if this form is for an update operation, `false` if for a create operation

Return type `boolean`

compliance: mandatory – This method must be implemented.

default_locale

Gets a default locale for `DisplayTexts` when a locale is not specified.

Returns the default locale

Return type `osid.locale.Locale`

compliance: mandatory – This method must be implemented.

locales

Gets a list of locales for available `DisplayText` translations that can be performed using this form.

Returns a list of available locales or an empty list if no translation operations are available

Return type `osid.locale.LocaleList`

compliance: mandatory – This method must be implemented.

set_locale(*language_type*, *script_type*)

Specifies a language and script type for `DisplayText` fields in this form.

Setting a locale to something other than the default locale may affect the `Metadata` in this form.

If multiple locales are available for managing translations, the `Metadata` indicates the fields are unset as they may be returning a defeult value based on the default locale.

Parameters

- **`language_type`** (`osid.type.Type`) – the language type
- **`script_type`** (`osid.type.Type`) – the script type

Raise `NullArgument` – `language_type` or `script_type` is null

Raise `Unsupported` – `language_type` and `script_type` not available from `get_locales()`

compliance: mandatory – This method must be implemented.

`journal_comment_metadata`

Gets the metadata for the comment corresponding to this form submission.

The comment is used for describing the nature of the change to the corresponding object for the purposes of logging and auditing.

Returns metadata for the comment

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`journal_comment`

`is_valid``()`

Tests if ths form is in a valid state for submission.

A form is valid if all required data has been supplied compliant with any constraints.

Returns `false` if there is a known error in this form, `true` otherwise

Return type `boolean`

Raise `OperationFailed` – attempt to perform validation failed

compliance: mandatory – This method must be implemented.

`validation_messages`

Gets text messages corresponding to additional instructions to pass form validation.

Returns a list of messages

Return type `osid.locale.DisplayText`

compliance: mandatory – This method must be implemented.

`invalid_metadata`

Gets a list of metadata for the elements in this form which are not valid.

Returns invalid metadata

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

Osid Identifiable Form

class `dlkit.osid.objects.`**`OsidIdentifiableForm`**

Bases: *`dlkit.osid.objects.OsidForm`*

The `OsidIdentifiableForm` is used to create and update identifiable objects.

The form is a container for data to be sent to an update or create method of a session.

Osid Extensible Form

class `dlkit.osid.objects.`**`OsidExtensibleForm`**

Bases: *`dlkit.osid.objects.OsidForm`*, *`dlkit.osid.markers.Extensible`*

The `OsidExtensibleForm` is used to create and update extensible objects.

The form is a container for data to be sent to an update or create method of a session.

`required_record_types`

Gets the required record types for this form.

The required records may change as a result of other data in this form and should be checked before submission.

Returns a list of required record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

Osid Browsable Form

class `dlkit.osid.objects.`**`OsidBrowsableForm`**

Bases: *`dlkit.osid.objects.OsidForm`*

The `OsidBrowsableForm` is used to create and update browsable objects.

The form is a container for data to be sent to an update or create method of a session.

Osid Temporal Form

class `dlkit.osid.objects.`**`OsidTemporalForm`**

Bases: *`dlkit.osid.objects.OsidForm`*

This form is used to create and update temporals.

`start_date_metadata`

Gets the metadata for a start date.

Returns metadata for the date

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`start_date`

`end_date_metadata`

Gets the metadata for an end date.

Returns metadata for the date

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`end_date`

Osid Subjugateable Form

class `dlkit.osid.objects.`**`OsidSubjugateableForm`**

Bases: *`dlkit.osid.objects.OsidForm`*

This form is used to create and update dependent objects.

Osid Aggregateable Form

class `dlkit.osid.objects.`**`OsidAggregateableForm`**

Bases: *`dlkit.osid.objects.OsidForm`*

This form is used to create and update assemblages.

Osid Containable Form

class `dlkit.osid.objects.`**`OsidContainableForm`**

Bases: *`dlkit.osid.objects.OsidForm`*

This form is used to create and update containers.

`sequestered_metadata`

Gets the metadata for the sequestered flag.

Returns metadata for the sequestered flag

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`sequestered`

Osid Sourceable Form

class `dlkit.osid.objects.`**`OsidSourceableForm`**

Bases: *`dlkit.osid.objects.OsidForm`*

This form is used to create and update sourceables.

`provider_metadata`

Gets the metadata for a provider.

Returns metadata for the provider

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`provider`

`branding_metadata`

Gets the metadata for the asset branding.

Returns metadata for the asset branding.

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`branding`

license_metadata

Gets the metadata for the license.

Returns metadata for the license

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

license_

Osid Federateable Form

class `dlkit.osid.objects.`**`OsidFederateableForm`**

Bases: *`dlkit.osid.objects.OsidForm`*

This form is used to create and update federateables.

Osid Operable Form

class `dlkit.osid.objects.`**`OsidOperableForm`**

Bases: *`dlkit.osid.objects.OsidForm`*

This form is used to create and update operables.

enabled_metadata

Gets the metadata for the enabled flag.

Returns metadata for the enabled flag

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

enabled

disabled_metadata

Gets the metadata for the disabled flag.

Returns metadata for the disabled flag

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

disabled

Osid Object Form

class `dlkit.osid.objects.`**`OsidObjectForm`**

Bases: *`dlkit.osid.objects.OsidIdentifiableForm`*, *`dlkit.osid.objects.OsidExtensibleForm`*, *`dlkit.osid.objects.OsidBrowsableForm`*

The `OsidObjectForm` is used to create and update `OsidObjects`.

The form is not an `OsidObject` but merely a container for data to be sent to an update or create method of a session. A provider may or may not combine the `OsidObject` and `OsidObjectForm` interfaces into a single object.

Generally, a set method parallels each get method of an `OsidObject`. Additionally, `Metadata` may be examined for each data element to assist in understanding particular rules concerning acceptable data.

The form may provide some feedback as to the validity of certain data updates before the update transaction is issued to the correspodning session but a successful modification of the form is not a guarantee of success for the update transaction. A consumer may elect to perform all updates within a single update transaction or break up a large update intio smaller units. The tradeoff is the granularity of error feedback vs. the performance gain of a single transaction.

As with all aspects of the OSIDs, nulls cannot be used. Methods to clear values are also defined in the form.

A new `OsidForm` should be acquired for each transaction upon an `OsidObject`. Forms should not be reused from one object to another even if the supplied data is the same as the forms may encapsulate data specific to the object requested. Example of changing a display name and a color defined in a color interface extension:

> ObjectForm form = session.getObjectFormForUpdate(objectId); form.setDisplayName("new name"); ColorForm recordForm = form.getFormRecord(colorRecordType); recordForm.setColor("green"); session.updateObject(objectId, form);

display_name_metadata

Gets the metadata for a display name.

Returns metadata for the display name

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

display_name

description_metadata

Gets the metadata for a description.

Returns metadata for the description

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

description

genus_type_metadata

Gets the metadata for a genus type.

Returns metadata for the genus

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

genus_type

Osid Relationship Form

class `dlkit.osid.objects.`**OsidRelationshipForm**

Bases: *`dlkit.osid.objects.OsidObjectForm`*, *`dlkit.osid.objects.OsidTemporalForm`*

This form is used to create and update relationshps.

Osid Catalog Form

class `dlkit.osid.objects.`**OsidCatalogForm**

Bases: *`dlkit.osid.objects.OsidObjectForm`*, *`dlkit.osid.objects.OsidSourceableForm`*, *`dlkit.osid.objects.OsidFederateableForm`*

This form is used to create and update catalogs.

Osid Rule Form

class `dlkit.osid.objects.`**`OsidRuleForm`**

Bases: *`dlkit.osid.objects.OsidObjectForm`*, *`dlkit.osid.objects.OsidOperableForm`*

This form is used to create and update rules.

`rule_metadata`

Gets the metadata for an associated rule.

Returns metadata for the rule

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`rule`

Osid Enabler Form

class `dlkit.osid.objects.`**`OsidEnablerForm`**

Bases: *`dlkit.osid.objects.OsidRuleForm`*, *`dlkit.osid.objects.OsidTemporalForm`*

This form is used to create and update enablers.

`schedule_metadata`

Gets the metadata for an associated schedule.

Returns metadata for the schedule

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`schedule`

`event_metadata`

Gets the metadata for an associated event.

Returns metadata for the event

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`event`

`cyclic_event_metadata`

Gets the metadata for the cyclic event.

Returns metadata for the cyclic event

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`cyclic_event`

`demographic_metadata`

Gets the metadata for an associated demographic.

Returns metadata for the resource.

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

demographic

Osid Constrainer Form

class `dlkit.osid.objects.`**`OsidConstrainerForm`**

Bases: *`dlkit.osid.objects.OsidRuleForm`*

This form is used to create and update constrainers.

Osid Processor Form

class `dlkit.osid.objects.`**`OsidProcessorForm`**

Bases: *`dlkit.osid.objects.OsidRuleForm`*

This form is used to create and update processors.

Osid Governator Form

class `dlkit.osid.objects.`**`OsidGovernatorForm`**

Bases: *`dlkit.osid.objects.OsidObjectForm`*, *`dlkit.osid.objects.OsidOperableForm`*, *`dlkit.osid.objects.OsidSourceableForm`*

This form is used to create and update governators.

Osid Compendium Form

class `dlkit.osid.objects.`**`OsidCompendiumForm`**

Bases: *`dlkit.osid.objects.OsidObjectForm`*, *`dlkit.osid.objects.OsidSubjugateableForm`*

This form is used to create and update governators.

start_date_metadata

Gets the metadata for a start date.

Returns metadata for the date

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

start_date

end_date_metadata

Gets the metadata for an end date.

Returns metadata for the date

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

end_date

interpolated_metadata

Gets the metadata for the interpolated flag.

Returns metadata for the interpolated flag

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

interpolated

extrapolated_metadata

Gets the metadata for the extrapolated flag.

Returns metadata for the extrapolated flag

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

extrapolated

Osid Capsule Form

class `dlkit.osid.objects.`**`OsidCapsuleForm`**

Bases: *`dlkit.osid.objects.OsidForm`*

This form is used to create and update capsules.

Osid List

class `dlkit.osid.objects.`**`OsidList`**

`OsidList` is the top-level interface for all OSID lists.

An OSID list provides sequential access, one at a time or many at a time, access to a set of elements. These elements are not required to be OsidObjects but generally are. The element retrieval methods are defined in the sub-interface of `OsidList` where the appropriate return type is defined.

Osid lists are a once pass through iteration of elements. The size of the object set and the means in which the element set is generated or stored is not known. Assumptions based on the length of the element set by copying the entire contents of the list into a fixed buffer should be done with caution a awareness that an implementation may return a number of elements ranging from zero to infinity.

Lists are returned by methods when multiple return values are possible. There is no guarantee that successive calls to the same method will return the same set of elements in a list. Unless an order is specified in an interface definition, the order of the elements is not known.

has_next()

Tests if there are more elements in this list.

Returns `true` if more elements are available in this list, `false` if the end of the list has been reached

Return type `boolean`

compliance: mandatory – This method must be implemented. implementation notes: Any errors that may result from accesing the underlying set of elements are to be deferred until the consumer attempts retrieval in which case the provider must return `true` for this method.

available()

Gets the number of elements available for retrieval.

The number returned by this method may be less than or equal to the total number of elements in this list. To determine if the end of the list has been reached, the method `has_next()` should be used. This method conveys what is known about the number of remaining elements at a point in time and can be used to determine a minimum size of the remaining elements, if known. A valid return is zero even if `has_next()` is true.

This method does not imply asynchronous usage. All OSID methods may block.

Returns the number of elements available for retrieval

Return type `cardinal`

compliance: mandatory – This method must be implemented. implementation notes: Any errors that may result from accesing the underlying set of elements are to be deferred until the consumer attempts retrieval in which case the provider must return a positive integer for this method so the consumer can continue execution to receive the error. In all other circumstances, the provider must not return a number greater than the number of elements known since this number will be fed as a parameter to the bulk retrieval method.

`skip`(*n*)

Skip the specified number of elements in the list.

If the number skipped is greater than the number of elements in the list, hasNext() becomes false and available() returns zero as there are no more elements to retrieve.

Parameters **`n`** (`cardinal`) – the number of elements to skip

compliance: mandatory – This method must be implemented.

Osid Node

class `dlkit.osid.objects.`**`OsidNode`**

Bases: *`dlkit.osid.markers.Identifiable`*, *`dlkit.osid.markers.Containable`*

A node interface for hierarchical objects.

The `Id` of the node is the `Id` of the object represented at this node.

`is_root`()

Tests if this node is a root in the hierarchy (has no parents).

A node may have no more parents available in this node structure but is not a root in the hierarchy. If both `is_root()` and `has_parents()` is false, the parents of this node may be accessed thorugh another node structure retrieval.

Returns `true` if this node is a root in the hierarchy, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`has_parents`()

Tests if any parents are available in this node structure.

There may be no more parents in this node structure however there may be parents that exist in the hierarchy.

Returns `true` if this node has parents, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`parent_ids`

Gets the parents of this node.

Returns the parents of this node

Return type `osid.id.IdList`

compliance: mandatory – This method must be implemented.

is_leaf()

Tests if this node is a leaf in the hierarchy (has no children).

A node may have no more children available in this node structure but is not a leaf in the hierarchy. If both `is_leaf()` and `has_children()` is false, the children of this node may be accessed thorugh another node structure retrieval.

Returns `true` if this node is a leaf in the hierarchy, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

has_children()

Tests if any children are available in this node structure.

There may be no more children available in this node structure but this node may have children in the hierarchy.

Returns `true` if this node has children, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

child_ids

Gets the children of this node.

Returns the children of this node

Return type `osid.id.IdList`

compliance: mandatory – This method must be implemented.

Markers

Osid Primitive

class `dlkit.osid.markers.`**OsidPrimitive**

A marker interface for an interface that behaves like a language primitive.

Primitive types, such as numbers and strings, do not encapsulate behaviors supplied by an OSID Provider. More complex primitives are expressed through interface definitions but are treated in a similar fashion as a language primitive. OSID Primitives supplied by an OSID Consumer must be consumable by any OSID Provider.

Identifiable

class `dlkit.osid.markers.`**Identifiable**

A marker interface for objects uniquely identified with an OSID `Id`.

ident

Gets the Id associated with this instance of this OSID object.

Persisting any reference to this object is done by persisting the Id returned from this method. The Id returned may be different than the Id used to query this object. In this case, the new Id should be preferred over the old one for future queries.

Returns the `Id`

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented. implementation notes: The `Id` is intended to be constant and persistent. A consumer may at any time persist the `Id` for retrieval at any future time. Ideally, the Id should consistently resolve into the designated object and not be reused. In cases where objects are deactivated after a certain lifetime the provider should endeavor not to obliterate the object or its `Id` but instead should update the properties of the object including the deactiavted status and the elimination of any unwanted pieces of data. As such, there is no means for updating an `Id` and providers should consider carefully the identification scheme to implement. `Id` assignments for objects are strictly in the realm of the provider and any errors should be fixed directly with the backend supporting system. Once an Id has been assigned in a production service it should be honored such that it may be necessary for the backend system to support Id aliasing to redirect the lookup to the current `Id`. Use of an Id OSID may be helpful to accomplish this task in a modular manner.

is_current()

Tests to see if the last method invoked retrieved up-to-date data.

Simple retrieval methods do not specify errors as, generally, the data is retrieved once at the time this object is instantiated. Some implementations may provide real-time data though the application may not always care. An implementation providing a real-time service may fall back to a previous snapshot in case of error. This method returns false if the data last retrieved was stale.

Returns `true` if the last data retrieval was up to date, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented. implementation notes: Providers should return false unless all getters are implemented using real-time queries, or some trigger process keeps the data in this object current. Providers should populate basic data elements at the time this object is instantiated, or set an error, to ensure some data availability.

Extensible

class `dlkit.osid.markers.`**Extensible**

A marker interface for objects that contain `OsidRecords`.

record_types

Gets the record types available in this object.

A record `Type` explicitly indicates the specification of an interface to the record. A record may or may not inherit other record interfaces through interface inheritance in which case support of a record type may not be explicit in the returned list. Interoperability with the typed interface to this object should be performed through `hasRecordType()`.

Returns the record types available

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

has_record_type(*record_type*)

Tests if this object supports the given record `Type`.

The given record type may be supported by the object through interface/type inheritence. This method should be checked before retrieving the record interface.

Parameters **record_type** (`osid.type.Type`) – a type

Returns `true` if a record of the given record `Type` is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

Browsable

class `dlkit.osid.markers.`**`Browsable`**

A marker interface for objects that offer property inspection.

`properties`

Gets a list of properties.

Properties provide a means for applications to display a representation of the contents of a record without understanding its `Type` specification. Applications needing to examine a specific property should use the extension interface defined by its `Type`.

Returns a list of properties

Return type `osid.PropertyList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – an authorization failure occurred

compliance: mandatory – This method must be implemented.

`get_properties_by_record_type``(`*record_type*`)`

Gets a list of properties corresponding to the specified record type.

Properties provide a means for applications to display a representation of the contents of a record without understanding its record interface specification. Applications needing to examine a specific propertyshould use the methods defined by the record `Type`. The resulting set includes properties specified by parents of the record `type` in the case a record's interface extends another.

Parameters **`record_type`** (`osid.type.Type`) – the record type corresponding to the properties set to retrieve

Returns a list of properties

Return type `osid.PropertyList`

Raise `NullArgument` – `record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – an authorization failure occurred

Raise `Unsupported` – `has_record_type(record_type)` is `false`

compliance: mandatory – This method must be implemented.

Suppliable

class `dlkit.osid.markers.`**`Suppliable`**

A marker interface for OSID Provider-owned objects used to supply input from an OSID Consumer.

Temporal

class `dlkit.osid.markers.`**`Temporal`**

`Temporal` is used to indicate the object endures for a period of time.

`is_effective``()`

Tests if the current date is within the start end end dates inclusive.

Returns `true` if this is effective, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`start_date`
Gets the start date.

Returns the start date

Return type `osid.calendaring.DateTime`

compliance: mandatory – This method must be implemented.

`end_date`
Gets the end date.

Returns the end date

Return type `osid.calendaring.DateTime`

compliance: mandatory – This method must be implemented.

Subjugateable

class `dlkit.osid.markers.`**`Subjugateable`**
A `Subjugateable` is an `OsidObject` dependent upon another `OsidObject`.

A `Subjugateable` is created in the context of the administering `OsidObject` that may not be reassigned.

A `Subjugateable` always has a fixed Id of it administering `OsidObject`.

Aggregateable

class `dlkit.osid.markers.`**`Aggregateable`**
`Aggregateable` is used for an `OsidObject` to indicate that some or all of the definition is based on an included set of other `OsidObjects` which are directly accessible and do not exist outside the context of the parent object.

`Aggregateables` allow for an `OsidObject` to stand alone without knowledge of the originating service.

An `Asset` is an example of an aggregate by including the `AssetContents`. An Asset also contains a provider however in this case the provider is categorized as a simple data attribute of the `Asset` that can be changed by updating the `Asset` using an `AssetForm`. The `AssetContent` differs in there exists a explicit mapping to the `Asset` managed through an `OsidSession` but accessible directly within the `Asset` to enable its consumption outside the Repository OSID.

This marker has little practicality other than to identify a service pattern that is neither a data attribute nor a separately accessible relationship or mapping.

Containable

class `dlkit.osid.markers.`**`Containable`**
A `Containable` is a kind of aggregate where an `OsidObject` is defined as a recursive composition of itself directly accessible without knowledge of the originating service.

`is_sequestered``()`
Tests if this `Containable` is sequestered in that it should not appear outside of its aggregated composition.

Returns `true` if this containable is sequestered, `false` if this containable may appear outside its aggregate

Return type `boolean`

compliance: mandatory – This method must be implemented.

Sourceable

class `dlkit.osid.markers.`**`Sourceable`**

`Sourceble` is used for `OsidObjects` where information about a provider is appropriate.

Examples of `Sourceables` are catalogs, compositions, and services.

`provider_id`

Gets the `Id` of the provider.

Returns the provider `Id`

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`provider`

Gets the `Resource` representing the provider.

Returns the provider

Return type `osid.resource.Resource`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`branding_ids`

Gets the branding asset `Ids`.

Returns a list of asset `Ids`

Return type `osid.id.IdList`

compliance: mandatory – This method must be implemented.

`branding`

Gets a branding, such as an image or logo, expressed using the `Asset` interface.

Returns a list of assets

Return type `osid.repository.AssetList`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`license_`

Gets the terms of usage.

An empty license means the terms are unknown.

Returns the license

Return type `osid.locale.DisplayText`

compliance: mandatory – This method must be implemented.

Federateable

class `dlkit.osid.markers.`**`Federateable`**

`Federateable` is used to indicate an `OsidObject` can be federated using the OSID Hierarchy pattern.

An OSID federation of `OsidObjects` is where it is inferred from the hiererarchy that any `OsidObject` "includes" its children.

Operable

class `dlkit.osid.markers.`**`Operable`**

`Operable` is used to indicate an `OsidObject` performs operations.

The active status indicates if the `Operable` is on or off. The active status is determined from the operational status and the enabling rules.

The operational status indicates the Operable is functioning. This status is not set administratively but instead refelects suitable conditions for operation.

Operables may be administratively turned on of off through the enabled and disabled administrative overrides. If there are no related `OsidEnabler` rules, then `is_enabled()` should be set to `true` and `is_disabled()` set to `false` for the `Operable` to be on and `is_enabled()` set to `false` and `is_disabled()` set to true for the `Operable` to be `off`. `is_enabled()` and `is_disabled()` cannot both be `tru` e.

If there are related `OsidEnabler` rules, the active status of at least one `OsidEnabler` results in a `true` value for `isOperational().` This active status can be overridden by setting `is_disabled()` to `true`. If there are no active `OsidEnabler` rules, `is_operational()` is false resulting in an `off Operable` unless `is_enabled()` is `true` .

For the active status to be completely determined by the `OsidEnablers, both is_enabled()` and `is_disabled()` should be `false` where the `is_active()` status is completely driven from `isOperational().`

`is_active``()`

Tests if this operable is active.

`is_active()` is `true` if `is_operational()` is `true` and `is_disabled()` is `false`, or `is_enabled()` is `true`.

Returns `true` if this operable is on, `false` if it is off

Return type `boolean`

compliance: mandatory – This method must be implemented.

`is_enabled``()`

Tests if this operable is administravely enabled.

Administratively enabling overrides any applied `OsidEnabler`. If this method returns `true` then `is_disabled()` must return `false`.

Returns `true` if this operable is enabled, `false` if the active status is determined by other rules

Return type `boolean`

compliance: mandatory – This method must be implemented.

`is_disabled``()`

Tests if this operable is administravely disabled.

Administratively disabling overrides any applied `OsidEnabler`. If this method returns `true` then `is_enabled()` must return `false`.

Returns `true` if this operable is disabled, `false` if the active status is determined by other rules

Return type `boolean`

compliance: mandatory – This method must be implemented.

`is_operational``()`

Tests if this `Operable` is operational.

This Operable is operational if any of the applied `OsidEnablers` are `true`.

Returns `true` if this operable is operational, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

Queries

Osid Query

class `dlkit.osid.queries.`**`OsidQuery`**

Bases: *`dlkit.osid.markers.Suppliable`*

The `OsidQuery` is used to assemble search queries.

An `OsidQuery` is available from an `OsidQuerySession` and defines methods to match objects. Once the desired parameters are set, the `OsidQuery` is given to the designated search method. The same `OsidQuery` returned from the session must be used in the search as the provider may utilize implementation-specific data wiithin the object.

If multiple data elements are set in this interface, the results matching all the given data (eg: AND) are returned.

Any match method inside an `OsidQuery` may be invoked multiple times. In the case of a match method, each invocation adds an element to an `OR` expression. Any of these terms may also be negated through the `match` flag.

OsidQuery { OsidQuery.matchDisplayName AND (OsidQuery.matchDescription OR OsidQuery.matchDescription)}

`OsidObjects` allow for the definition of an additonal records and the `OsidQuery` parallels this mechanism. An interface type of an `OsidObject` record must also define the corresponding `OsidQuery` record which is available through query interfaces. Multiple requests of these typed interfaces may return the same underlying object and thus it is only useful to request once.

An `OsidQuery` may be used to query for set or unset values using the “match any” methods. A field that has not bee explicitly assigned may default to a value. If multiple language translations exist and the query session is placed in a non-default locale, fields that have not been explicitly assigned in the non-default locale are considered unset even if the values from the default locale appear in the objects.

`string_match_types`

Gets the string matching types supported.

A string match type specifies the syntax of the string query, such as matching a word or including a wildcard or regular expression.

Returns a list containing the supported string match types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

`supports_string_match_type`(*string_match_type*)

Tests if the given string matching type is supported.

Parameters **`string_match_type`** (`osid.type.Type`) – a `Type` indicating a string match type

Returns `true` if the given Type is supported, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `string_match_type` is `null`

compliance: mandatory – This method must be implemented.

`match_keyword`(*keyword*, *string_match_type*, *match*)

Adds a keyword to match.

Multiple keywords can be added to perform a boolean `OR` among them. A keyword may be applied to any of the elements defined in this object such as the display name, description or any method defined in an interface implemented by this object.

Parameters

- **`keyword`** (`string`) – keyword to match
- **`string_match_type`** (`osid.type.Type`) – the string match type
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `keyword` is not of `string_match_type`

Raise `NullArgument` – `keyword` or `string_match_type` is `null`

Raise `Unsupported` – `supports_string_match_type(string_match_type)` is `false`

compliance: mandatory – This method must be implemented.

`keyword_terms`

`match_any`(*match*)

Matches any object.

Parameters **`match`** (`boolean`) – `true` to match any object , `false` to match no objects

compliance: mandatory – This method must be implemented.

`any_terms`

Osid Identifiable Query

class `dlkit.osid.queries.`**`OsidIdentifiableQuery`**

Bases: *`dlkit.osid.queries.OsidQuery`*

The `OsidIdentiableQuery` is used to assemble search queries for `Identifiable` objects.

An `OsidIdentifiableQuery` is available from an `OsidQuerySession` and defines methods to match objects. Once the desired parameters are set, the `OsidIdentifiableQuery` is given to the designated search method. The same `OsidIdentifiableQuery` returned from the session must be used in the search as the provider may utilize implementation-specific data wiithin the object.

If multiple data elements are set in this interface, the results matching all the given data (eg: AND) are returned.

match_id(*id_*, *match*)

Adds an `Id` to match.

Multiple `Ids` can be added to perform a boolean `OR` among them.

Parameters

- **id** (`osid.id.Id`) – `Id` to match
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `id` is `null`

compliance: mandatory – This method must be implemented.

id_terms

Osid Extensible Query

class `dlkit.osid.queries.`**OsidExtensibleQuery**

Bases: *dlkit.osid.queries.OsidQuery*, *dlkit.osid.markers.Extensible*

The `OsidExtensibleQuery` is used to assemble search queries for `Extensible` objects.

An `OsidExtensibleQuery` is available from an `OsidQuerySession` and defines methods to match objects. Once the desired parameters are set, the `OsidExtensibleQuery` is given to the designated search method. The same `OsidExtensibleQuery` returned from the session must be used in the search as the provider may utilize implementation-specific data wiithin the object.

If multiple data elements are set in this interface, the results matching all the given data (eg: AND) are returned.

match_record_type(*record_type*, *match*)

Sets a `Type` for querying objects having records implementing a given record type.

Parameters

- **record_type** (`osid.type.Type`) – a record type
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `record_type` is `null`

compliance: mandatory – This method must be implemented.

match_any_record(*match*)

Matches an object that has any record.

Parameters **match** (`boolean`) – `true` to match any record, `false` to match objects with no records

compliance: mandatory – This method must be implemented.

record_terms

Osid Browsable Query

class `dlkit.osid.queries.`**OsidBrowsableQuery**

Bases: *dlkit.osid.queries.OsidQuery*

The `OsidBrowsableQuery` is used to assemble search queries for `Browsable` objects.

An `OsidBrowsableQuery` is available from an `OsidQuerySession` and defines methods to match objects. Once the desired parameters are set, the `OsidBrowsableQuery` is given to the designated search method. The same `OsidBrowsableQuery` returned from the session must be used in the search as the provider may utilize implementation-specific data wiithin the object.

If multiple data elements are set in this interface, the results matching all the given data (eg: AND) are returned.

Osid Temporal Query

class `dlkit.osid.queries.`**`OsidTemporalQuery`**

Bases: *`dlkit.osid.queries.OsidQuery`*

This is the query interface for searching temporal objects.

Each method specifies an `AND` term while multiple invocations of the same method produce a nested `OR`.

`match_effective`(*match*)

Match effective objects where the current date falls within the start and end dates inclusive.

Parameters **`match`** (`boolean`) – `true` to match any effective, `false` to match ineffective

compliance: mandatory – This method must be implemented.

`effective_terms`

`match_start_date`(*start*, *end*, *match*)

Matches temporals whose start date falls in between the given dates inclusive.

Parameters

- **`start`** (`osid.calendaring.DateTime`) – start of date range
- **`end`** (`osid.calendaring.DateTime`) – end of date range
- **`match`** (`boolean`) – `true` if a positive match, `false` for a negative match

Raise `InvalidArgument` – `start` is less than `end`

Raise `NullArgument` – `start` or `end` is `null`

compliance: mandatory – This method must be implemented.

`match_any_start_date`(*match*)

Matches temporals with any start date set.

Parameters **`match`** (`boolean`) – `true` to match any start date, `false` to match no start date

compliance: mandatory – This method must be implemented.

`start_date_terms`

`match_end_date`(*start*, *end*, *match*)

Matches temporals whose effective end date falls in between the given dates inclusive.

Parameters

- **`start`** (`osid.calendaring.DateTime`) – start of date range
- **`end`** (`osid.calendaring.DateTime`) – end of date range
- **`match`** (`boolean`) – `true` if a positive match, `false` for negative match

Raise `InvalidArgument` – `start` is less than `end`

Raise `NullArgument` – `start` or `end` is `null`

compliance: mandatory – This method must be implemented.

match_any_end_date(*match*)

Matches temporals with any end date set.

Parameters match (boolean) – true to match any end date, false to match no start date

compliance: mandatory – This method must be implemented.

end_date_terms

match_date(*from_*, *to*, *match*)

Matches temporals where the given date range falls entirely between the start and end dates inclusive.

Parameters

- **from** (osid.calendaring.DateTime) – start date
- **to** (osid.calendaring.DateTime) – end date
- **match** (boolean) – true if a positive match, false for a negative match

Raise InvalidArgument – from is less than to

Raise NullArgument – from or to is null

compliance: mandatory – This method must be implemented.

date_terms

Osid Subjugateable Query

class dlkit.osid.queries.**OsidSubjugateableQuery**

Bases: *dlkit.osid.queries.OsidQuery*

The OsidSubjugateableQuery is used to assemble search queries for dependent objects.

Osid Aggregateable Query

class dlkit.osid.queries.**OsidAggregateableQuery**

Bases: *dlkit.osid.queries.OsidQuery*

The OsidAggregateableQuery is used to assemble search queries for assemblages.

Osid Containable Query

class dlkit.osid.queries.**OsidContainableQuery**

Bases: *dlkit.osid.queries.OsidQuery*

This is the query interface for searching containers.

Each method specifies an AND term while multiple invocations of the same method produce a nested OR.

match_sequestered(*match*)

Match containables that are sequestered.

Parameters match (boolean) – true to match any sequestered containables, false to match non-sequestered containables

compliance: mandatory – This method must be implemented.

sequestered_terms

Osid Sourceable Query

class dlkit.osid.queries.**OsidSourceableQuery**

Bases: *dlkit.osid.queries.OsidQuery*

The OsidSourceableQuery is used to assemble search queries for sourceables.

match_provider_id(*resource_id*, *match*)

Match the Id of the provider resource.

Parameters

- **resource_id** (osid.id.Id) – Id to match
- **match** (boolean) – true if for a positive match, false for a negative match

Raise NullArgument – resource_id is null

compliance: mandatory – This method must be implemented.

provider_id_terms

supports_provider_query()

Tests if a ResourceQuery for the provider is available.

Returns true if a resource query is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

get_provider_query(*match*)

Gets the query for the provider.

Each retrieval performs a boolean OR.

Parameters match (boolean) – true if for a positive match, false for a negative match

Returns the provider query

Return type osid.resource.ResourceQuery

Raise Unimplemented – supports_provider_query() is false

compliance: optional – This method must be implemented if ‘‘supports_provider_query()‘‘ is ‘‘true‘‘.

match_any_provider(*match*)

Match sourceables with a provider value.

Parameters match (boolean) – true to match sourceables with any provider, false to match sourceables with no providers

compliance: mandatory – This method must be implemented.

provider_terms

match_branding_id(*asset_id*, *match*)

Match the Id of an asset used for branding.

Parameters

- **asset_id** (osid.id.Id) – Id to match
- **match** (boolean) – true if for a positive match, false for a negative match

Raise NullArgument – asset_id is null

compliance: mandatory – This method must be implemented.

`branding_id_terms`

`supports_branding_query``()`
Tests if an `AssetQuery` for the branding is available.

Returns `true` if a asset query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`get_branding_query``(`*match*`)`
Gets the query for an asset.

Each retrieval performs a boolean `OR`.

Parameters **`match`** (`boolean`) – `true` if for a positive match, `false` for a negative match

Returns the asset query

Return type `osid.repository.AssetQuery`

Raise `Unimplemented` – `supports_branding_query()` is `false`

compliance: optional – This method must be implemented if ``supports_branding_query()`` is ``true``.

`match_any_branding``(`*match*`)`
Match sourceables with any branding.

Parameters **`match`** (`boolean`) – `true` to match any asset, `false` to match no assets

compliance: mandatory – This method must be implemented.

`branding_terms`

`match_license``(`*license_*, *string_match_type*, *match*`)`
Adds a license to match.

Multiple license matches can be added to perform a boolean `OR` among them.

Parameters

- **`license`** (`string`) – a string to match
- **`string_match_type`** (`osid.type.Type`) – the string match type
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `license` is not of `string_match_type`

Raise `NullArgument` – `license` or `string_match_type` is `null`

Raise `Unsupported` – `supports_string_match_type(string_match_type)` is `false`

compliance: mandatory – This method must be implemented.

`match_any_license``(`*match*`)`
Matches any object with a license.

Parameters **`match`** (`boolean`) – `true` to match any license, `false` to match objects with no license

compliance: mandatory – This method must be implemented.

`license_terms`

Osid Federateable Query

class dlkit.osid.queries.**OsidFederateableQuery**

Bases: *dlkit.osid.queries.OsidQuery*

The OsidFederateableQuery is used to assemble search queries for federated objects.

Osid Operable Query

class dlkit.osid.queries.**OsidOperableQuery**

Bases: *dlkit.osid.queries.OsidQuery*

This is the query interface for searching operables.

Each method specifies an AND term while multiple invocations of the same method produce a nested OR.

match_active(*match*)

Matches active.

Parameters match (boolean) – true to match active, false to match inactive

compliance: mandatory – This method must be implemented.

active_terms

match_enabled(*match*)

Matches administratively enabled.

Parameters match (boolean) – true to match administratively enabled, false otherwise

compliance: mandatory – This method must be implemented.

enabled_terms

match_disabled(*match*)

Matches administratively disabled.

Parameters match (boolean) – true to match administratively disabled, false otherwise

compliance: mandatory – This method must be implemented.

disabled_terms

match_operational(*match*)

Matches operational operables.

Parameters match (boolean) – true to match operational, false to match not operational

compliance: mandatory – This method must be implemented.

operational_terms

Osid Object Query

class dlkit.osid.queries.**OsidObjectQuery**

Bases: *dlkit.osid.queries.OsidIdentifiableQuery*, *dlkit.osid.queries.OsidExtensibleQuery*, *dlkit.osid.queries.OsidBrowsableQuery*

The OsidObjectQuery is used to assemble search queries.

An OsidObjectQuery is available from an OsidSession and defines methods to query for an OsidObject that includes setting a display name and a description. Once the desired parameters are set, the `OsidQuery` is given to the designated search method. The same `OsidQuery` returned from the session must be used in the search as the provider may utilize implementation-specific data wiithin the object.

If multiple data elements are set in this interface, the results matching all the given data (eg: AND) are returned.

Any match method inside an `OsidObjectQuery` may be invoked multiple times. In the case of a match method, each invocation adds an element to an `OR` expression. Any of these terms may also be negated through the `match` flag.

> OsidObjectQuery { OsidQuery.matchDisplayName AND (OsidQuery.matchDescription OR OsidObjectQuery.matchDescription)}

`OsidObjects` allow for the definition of an additonal records and the `OsidQuery` parallels this mechanism. An interface type of an `OsidObject` record must also define the corresponding `OsidQuery` record which is available through query interfaces. Multiple requests of these typed interfaces may return the same underlying object and thus it is only useful to request once.

String searches are described using a string search `Type` that indicates the type of regular expression or wildcarding encoding. Compatibility with a strings search `Type` can be tested within this interface.

As with all aspects of OSIDs, nulls cannot be used. Separate tests are available for querying for unset values except for required fields.

An example to find all objects whose name starts with "Fred" or whose name starts with "Barney", but the word "dinosaur" does not appear in the description and not the color is not purple. `ColorQuery` is a record of the object that defines a color.

> ObjectObjectQuery query; query = session.getObjectQuery(); query.matchDisplayName("Fred*", wildcardStringMatchType, true); query.matchDisplayName("Barney*", wildcardStringMatchType, true); query.matchDescriptionMatch("dinosaur", wordStringMatchType, false);
>
> ColorQuery recordQuery; recordQuery = query.getObjectRecord(colorRecordType); recordQuery.matchColor("purple", false); ObjectList list = session.getObjectsByQuery(query);

`match_display_name`(*display_name*, *string_match_type*, *match*)

Adds a display name to match.

Multiple display name matches can be added to perform a boolean `OR` among them.

Parameters

- **`display_name`** (`string`) – display name to match
- **`string_match_type`** (`osid.type.Type`) – the string match type
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `display_name` is not of `string_match_type`

Raise `NullArgument` – `display_name` or `string_match_type` is `null`

Raise `Unsupported` – `supports_string_match_type(string_match_type)` is `false`

compliance: mandatory – This method must be implemented.

`match_any_display_name`(*match*)

Matches any object with a display name.

Parameters **`match`** (`boolean`) – `true` to match any display name, `false` to match objects with no display name

compliance: mandatory – This method must be implemented.

`display_name_terms`

`match_description`(*description*, *string_match_type*, *match*)

Adds a description name to match.

Multiple description matches can be added to perform a boolean `OR` among them.

Parameters

- **`description`** (`string`) – description to match
- **`string_match_type`** (`osid.type.Type`) – the string match type
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `description` is not of `string_match_type`

Raise `NullArgument` – `description` or `string_match_type` is `null`

Raise `Unsupported` – `supports_string_match_type(string_match_type)` is `false`

compliance: mandatory – This method must be implemented.

`match_any_description`(*match*)

Matches a description that has any value.

Parameters **`match`** (`boolean`) – `true` to match any description, `false` to match descriptions with no values

compliance: mandatory – This method must be implemented.

`description_terms`

`match_genus_type`(*genus_type*, *match*)

Sets a `Type` for querying objects of a given genus.

A genus type matches if the specified type is the same genus as the object genus type.

Parameters

- **`genus_type`** (`osid.type.Type`) – the object genus type
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `genus_type` is `null`

compliance: mandatory – This method must be implemented.

`match_any_genus_type`(*match*)

Matches an object that has any genus type.

Parameters **`match`** (`boolean`) – `true` to match any genus type, `false` to match objects with no genus type

compliance: mandatory – This method must be implemented.

`genus_type_terms`

`match_parent_genus_type`(*genus_type*, *match*)

Sets a `Type` for querying objects of a given genus.

A genus type matches if the specified type is the same genus as the object or if the specified type is an ancestor of the object genus in a type hierarchy.

Parameters

- **`genus_type`** (`osid.type.Type`) – the object genus type
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `genus_type` is `null`

compliance: mandatory – This method must be implemented.

`parent_genus_type_terms`

`match_subject_id`(*subject_id*, *match*)
Matches an object with a relationship to the given subject.

Parameters

- **subject_id** (`osid.id.Id`) – a subject `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `subject_id` is `null`

compliance: mandatory – This method must be implemented.

`subject_id_terms`

`supports_subject_query`()
Tests if a `SubjectQuery` is available.

Returns `true` if a subject query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`subject_query`
Gets the query for a subject.

Multiple retrievals produce a nested `OR` term.

Returns the subject query

Return type `osid.ontology.SubjectQuery`

Raise `Unimplemented` – `supports_subject_query()` is `false`

compliance: optional – This method must be implemented if ``supports_subject_query()`` is ``true``.

`match_any_subject`(*match*)
Matches an object that has any relationship to a `Subject`.

Parameters match (`boolean`) – `true` to match any subject, `false` to match objects with no subjects

compliance: mandatory – This method must be implemented.

`subject_terms`

`supports_subject_relevancy_query`()
Tests if a `RelevancyQuery` is available to provide queries about the relationships to `Subjects`.

Returns `true` if a relevancy entry query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`subject_relevancy_query`
Gets the query for a subject relevancy.

Multiple retrievals produce a nested `OR` term.

Returns the relevancy query

Return type `osid.ontology.RelevancyQuery`

Raise `Unimplemented` – `supports_subject_relevancy_query()` is `false`

compliance: optional – This method must be implemented if ``supports_subject_relevancy_query()`` is ``true``.

subject_relevancy_terms

match_state_id(*state_id*, *match*)
Matches an object mapped to the given state.

Parameters

- **state_id** (`osid.id.Id`) – a state `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `state_id` is `null`

compliance: mandatory – This method must be implemented.

state_id_terms

supports_state_query()
Tests if a `StateQuery` is available to provide queries of processed objects.

Returns `true` if a state query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

state_query
Gets the query for a state.

Multiple retrievals produce a nested `OR` term.

Returns the journal entry query

Return type `osid.process.StateQuery`

Raise `Unimplemented` – `supports_state_query()` is `false`

compliance: optional – This method must be implemented if ``supports_state_query()`` is ``true``.

match_any_state(*match*)
Matches an object that has any mapping to a `State` in the given `Process`.

Parameters **match** (`boolean`) – `true` to match any state, `false` to match objects with no states

compliance: mandatory – This method must be implemented.

state_terms

match_comment_id(*comment_id*, *match*)
Matches an object that has the given comment.

Parameters

- **comment_id** (`osid.id.Id`) – a comment `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `comment_id` is `null`

compliance: mandatory – This method must be implemented.

comment_id_terms

supports_comment_query()

Tests if a `CommentQuery` is available.

Returns `true` if a comment query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

comment_query

Gets the query for a comment.

Multiple retrievals produce a nested `OR` term.

Returns the comment query

Return type `osid.commenting.CommentQuery`

Raise `Unimplemented` – `supports_comment_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_comment_query()‘‘ is ‘‘true‘‘.

match_any_comment(*match*)

Matches an object that has any `Comment` in the given `Book`.

Parameters **match** (`boolean`) – `true` to match any comment, `false` to match objects with no comments

compliance: mandatory – This method must be implemented.

comment_terms

match_journal_entry_id(*journal_entry_id*, *match*)

Matches an object that has the given journal entry.

Parameters

- **journal_entry_id** (`osid.id.Id`) – a journal entry `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `journal_entry_id` is `null`

compliance: mandatory – This method must be implemented.

journal_entry_id_terms

supports_journal_entry_query()

Tests if a `JournalEntry` is available to provide queries of journaled `OsidObjects`.

Returns `true` if a journal entry query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

journal_entry_query

Gets the query for a journal entry.

Multiple retrievals produce a nested `OR` term.

Returns the journal entry query

Return type `osid.journaling.JournalEntryQuery`

Raise `Unimplemented` – `supports_journal_entry_query()` is `false` *compliance: optional – This method must be implemented if ``supports_journal_entry_query()`` is ``true``.*

`match_any_journal_entry`(*match*)
Matches an object that has any `JournalEntry` in the given `Journal`.

Parameters **`match`** (`boolean`) – `true` to match any journal entry, `false` to match objects with no journal entries

compliance: mandatory – This method must be implemented.

`journal_entry_terms`

`supports_statistic_query`()
Tests if a `StatisticQuery` is available to provide statistical queries.

Returns `true` if a statistic query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`statistic_query`
Gets the query for a statistic.

Multiple retrievals produce a nested `OR` term.

Returns the statistic query

Return type `osid.metering.StatisticQuery`

Raise `Unimplemented` – `supports_statistic_query()` is `false`

compliance: optional – This method must be implemented if ``supports_statistic_query()`` is ``true``.

`match_any_statistic`(*match*)
Matches an object that has any `Statistic`.

Parameters **`match`** (`boolean`) – `true` to match any statistic, `false` to match objects with no statistics

compliance: mandatory – This method must be implemented.

`statistic_terms`

`match_credit_id`(*credit_id*, *match*)
Matches an object that has the given credit.

Parameters

- **`credit_id`** (`osid.id.Id`) – a credit `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `credit_id` is `null`

compliance: mandatory – This method must be implemented.

`credit_id_terms`

`supports_credit_query`()
Tests if a `CreditQuery` is available to provide queries of related acknowledgements.

Returns `true` if a credit query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`credit_query`

Gets the query for an ackowledgement credit.

Multiple retrievals produce a nested `OR` term.

Returns the credit query

Return type `osid.acknowledgement.CreditQuery`

Raise `Unimplemented` – `supports_credit_query()` is `false`

compliance: optional – This method must be implemented if ``supports_credit_query()`` is ``true``.

`match_any_credit`(*match*)

Matches an object that has any `Credit`.

Parameters **`match`** (`boolean`) – `true` to match any credit, `false` to match objects with no credits

compliance: mandatory – This method must be implemented.

`credit_terms`

`match_relationship_id`(*relationship_id*, *match*)

Matches an object that has the given relationship.

Parameters

- **`relationship_id`** (`osid.id.Id`) – a relationship `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `relationship_id` is `null`

compliance: mandatory – This method must be implemented.

`relationship_id_terms`

`supports_relationship_query`()

Tests if a `RelationshipQuery` is available.

Returns `true` if a relationship query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`relationship_query`

Gets the query for relationship.

Multiple retrievals produce a nested `OR` term.

Returns the relationship query

Return type `osid.relationship.RelationshipQuery`

Raise `Unimplemented` – `supports_relationship_query()` is `false`

compliance: optional – This method must be implemented if ``supports_relationship_query()`` is ``true``.

`match_any_relationship`(*match*)

Matches an object that has any `Relationship`.

Parameters **`match`** (`boolean`) – `true` to match any relationship, `false` to match objects with no relationships

compliance: mandatory – This method must be implemented.

`relationship_terms`

match_relationship_peer_id(*peer_id*, *match*)
Matches an object that has a relationship to the given peer `Id`.

Parameters

- **peer_id** (`osid.id.Id`) – a relationship peer `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `peer_id` is `null`

compliance: mandatory – This method must be implemented.

relationship_peer_id_terms

Osid Relationship Query

class `dlkit.osid.queries.`**OsidRelationshipQuery**
Bases: *dlkit.osid.queries.OsidObjectQuery*, *dlkit.osid.queries.OsidTemporalQuery*

This is the query interface for searching relationships.

Each method specifies an `AND` term while multiple invocations of the same method produce a nested `OR`.

match_end_reason_id(*state_id*, *match*)
Match the `Id` of the end reason state.

Parameters

- **state_id** (`osid.id.Id`) – `Id` to match
- **match** (`boolean`) – `true` if for a positive match, `false` for a negative match

Raise `NullArgument` – `rule_id` is `null`

compliance: mandatory – This method must be implemented.

end_reason_id_terms

supports_end_reason_query()
Tests if a `StateQuery` for the end reason is available.

Returns `true` if a end reason query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

get_end_reason_query(*match*)
Gets the query for the end reason state.

Each retrieval performs a boolean `OR`.

Parameters **match** (`boolean`) – `true` if for a positive match, `false` for a negative match

Returns the state query

Return type `osid.process.StateQuery`

Raise `Unimplemented` – `supports_end_reason_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_end_reason_query()‘‘ is ‘‘true‘‘.

match_any_end_reason(*match*)
Match any end reason state.

Parameters **`match`** (`boolean`) – `true` to match any state, `false` to match no state

compliance: mandatory – This method must be implemented.

`end_reason_terms`

Osid Catalog Query

class `dlkit.osid.queries.`**`OsidCatalogQuery`**

Bases: *`dlkit.osid.queries.OsidObjectQuery`*, *`dlkit.osid.queries.OsidSourceableQuery`*, *`dlkit.osid.queries.OsidFederateableQuery`*

The `OsidCatalogQuery` is used to assemble search queries for catalogs.

Osid Rule Query

class `dlkit.osid.queries.`**`OsidRuleQuery`**

Bases: *`dlkit.osid.queries.OsidObjectQuery`*, *`dlkit.osid.queries.OsidOperableQuery`*

This is the query interface for searching rules.

Each method specifies an `AND` term while multiple invocations of the same method produce a nested `OR`.

`match_rule_id`(*rule_id*, *match*)

Match the `Id` of the rule.

Parameters

- **`rule_id`** (`osid.id.Id`) – `Id` to match
- **`match`** (`boolean`) – `true` if for a positive match, `false` for a negative match

Raise `NullArgument` – `rule_id` is `null`

compliance: mandatory – This method must be implemented.

`rule_id_terms`

`supports_rule_query`()

Tests if a `RuleQuery` for the rule is available.

Returns `true` if a rule query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`get_rule_query`(*match*)

Gets the query for the rule.

Each retrieval performs a boolean `OR`.

Parameters **`match`** (`boolean`) – `true` if for a positive match, `false` for a negative match

Returns the rule query

Return type `osid.rules.RuleQuery`

Raise `Unimplemented` – `supports_rule_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_rule_query()‘‘ is ‘‘true‘‘.

`match_any_rule`(*match*)

Match any associated rule.

Parameters **match** (boolean) – true to match any rule, false to match no rules

compliance: mandatory – This method must be implemented.

rule_terms

Osid Enabler Query

class dlkit.osid.queries.**OsidEnablerQuery**

Bases: *dlkit.osid.queries.OsidRuleQuery*, *dlkit.osid.queries.OsidTemporalQuery*

This is the query interface for searching enablers.

Each method specifies an AND term while multiple invocations of the same method produce a nested OR.

match_schedule_id(*schedule_id*, *match*)

Match the Id of an associated schedule.

Parameters

- **schedule_id** (osid.id.Id) – Id to match
- **match** (boolean) – true if for a positive match, false for a negative match

Raise NullArgument – schedule_id is null

compliance: mandatory – This method must be implemented.

schedule_id_terms

supports_schedule_query()

Tests if a ScheduleQuery for the rule is available.

Returns true if a schedule query is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

get_schedule_query(*match*)

Gets the query for the schedule.

Each retrieval performs a boolean OR.

Parameters **match** (boolean) – true if for a positive match, false for a negative match

Returns the schedule query

Return type osid.calendaring.ScheduleQuery

Raise Unimplemented – supports_schedule_query() is false

compliance: optional – This method must be implemented if ‘‘supports_schedule_query()‘‘ is ‘‘true‘‘.

match_any_schedule(*match*)

Match any associated schedule.

Parameters **match** (boolean) – true to match any schedule, false to match no schedules

compliance: mandatory – This method must be implemented.

schedule_terms

match_event_id(*event_id*, *match*)

Match the Id of an associated event.

Parameters

- **event_id** (osid.id.Id) – Id to match
- **match** (boolean) – true if for a positive match, false for a negative match

Raise NullArgument – event_id is null

compliance: mandatory – This method must be implemented.

event_id_terms

supports_event_query()

Tests if a EventQuery for the rule is available.

Returns true if an event query is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

get_event_query(*match*)

Gets the query for the event.

Each retrieval performs a boolean OR.

Parameters **match** (boolean) – true if for a positive match, false for a negative match

Returns the event query

Return type osid.calendaring.EventQuery

Raise Unimplemented – supports_event_query() is false

compliance: optional – This method must be implemented if ‘‘supports_event_query()‘‘ is ‘‘true‘‘.

match_any_event(*match*)

Match any associated event.

Parameters **match** (boolean) – true to match any event, false to match no events

compliance: mandatory – This method must be implemented.

event_terms

match_cyclic_event_id(*cyclic_event_id*, *match*)

Sets the cyclic event Id for this query.

Parameters

- **cyclic_event_id** (osid.id.Id) – the cyclic event Id
- **match** (boolean) – true for a positive match, false for a negative match

Raise NullArgument – cyclic_event_id is null

compliance: mandatory – This method must be implemented.

cyclic_event_id_terms

supports_cyclic_event_query()

Tests if a CyclicEventQuery is available.

Returns true if a cyclic event query is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

cyclic_event_query
Gets the query for a cyclic event.

Multiple retrievals produce a nested `OR` term.

Returns the cyclic event query

Return type `osid.calendaring.cycle.CyclicEventQuery`

Raise `Unimplemented` – `supports_cyclic_event_query()` is `false`

compliance: optional – This method must be implemented if ``supports_cyclic_event_query()`` is ``true``.

match_any_cyclic_event(*match*)
Matches any enabler with a cyclic event.

Parameters **match** (`boolean`) – `true` to match any enablers with a cyclic event, `false` to match enablers with no cyclic events

compliance: mandatory – This method must be implemented.

cyclic_event_terms

match_demographic_id(*resource_id*, *match*)
Match the `Id` of the demographic resource.

Parameters

- **resource_id** (`osid.id.Id`) – `Id` to match
- **match** (`boolean`) – `true` if for a positive match, `false` for a negative match

Raise `NullArgument` – `resource_id` is `null`

compliance: mandatory – This method must be implemented.

demographic_id_terms

supports_demographic_query()
Tests if a `ResourceQuery` for the demographic is available.

Returns `true` if a resource query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

get_demographic_query(*match*)
Gets the query for the resource.

Each retrieval performs a boolean `OR`.

Parameters **match** (`boolean`) – `true` if for a positive match, `false` for a negative match

Returns the resource query

Return type `osid.resource.ResourceQuery`

Raise `Unimplemented` – `supports_resource_query()` is `false`

compliance: optional – This method must be implemented if ``supports_resource_query()`` is ``true``.

match_any_demographic(*match*)
Match any associated resource.

Parameters **match** (`boolean`) – `true` to match any demographic, `false` to match no rules

compliance: mandatory – This method must be implemented.

demographic_terms

Osid Constrainer Query

class `dlkit.osid.queries.`**`OsidConstrainerQuery`**

Bases: *`dlkit.osid.queries.OsidRuleQuery`*

This is the query interface for searching constrainers.

Each method specifies an `AND` term while multiple invocations of the same method produce a nested `OR`.

Osid Processor Query

class `dlkit.osid.queries.`**`OsidProcessorQuery`**

Bases: *`dlkit.osid.queries.OsidRuleQuery`*

This is the query interface for searching processors.

Each method specifies an `AND` term while multiple invocations of the same method produce a nested `OR`.

Osid Governator Query

class `dlkit.osid.queries.`**`OsidGovernatorQuery`**

Bases: *`dlkit.osid.queries.OsidObjectQuery`*, *`dlkit.osid.queries.OsidOperableQuery`*, *`dlkit.osid.queries.OsidSourceableQuery`*

This is the query interface for searching governers.

Each method specifies an `AND` term while multiple invocations of the same method produce a nested `OR`.

Osid Compendium Query

class `dlkit.osid.queries.`**`OsidCompendiumQuery`**

Bases: *`dlkit.osid.queries.OsidObjectQuery`*, *`dlkit.osid.queries.OsidSubjugateableQuery`*

This is the query interface for searching reports.

Each method specifies an `AND` term while multiple invocations of the same method produce a nested `OR`.

`match_start_date`(*start*, *end*, *match*)

Matches reports whose start date falls in between the given dates inclusive.

Parameters

- **start** (`osid.calendaring.DateTime`) – start of date range
- **end** (`osid.calendaring.DateTime`) – end of date range
- **match** (`boolean`) – `true` if a positive match, `false` for a negative match

Raise `InvalidArgument` – `start` is less than `end`

Raise `NullArgument` – `start` or `end` is `null`

compliance: mandatory – This method must be implemented.

`match_any_start_date`(*match*)

Matches reports with any start date set.

Parameters **`match`** (`boolean`) – `true` to match any start date, `false` to match no start date

compliance: mandatory – This method must be implemented.

`start_date_terms`

`match_end_date`(*start*, *end*, *match*)

Matches reports whose effective end date falls in between the given dates inclusive.

Parameters

- **`start`** (`osid.calendaring.DateTime`) – start of date range
- **`end`** (`osid.calendaring.DateTime`) – end of date range
- **`match`** (`boolean`) – `true` if a positive match, `false` for negative match

Raise `InvalidArgument` – `start` is less than `end`

Raise `NullArgument` – `start` or `end` is `null`

compliance: mandatory – This method must be implemented.

`match_any_end_date`(*match*)

Matches reports with any end date set.

Parameters **`match`** (`boolean`) – `true` to match any end date, `false` to match no start date

compliance: mandatory – This method must be implemented.

`end_date_terms`

`match_interpolated`(*match*)

Match reports that are interpolated.

Parameters **`match`** (`boolean`) – `true` to match any interpolated reports, `false` to match non-interpolated reports

compliance: mandatory – This method must be implemented.

`interpolated_terms`

`match_extrapolated`(*match*)

Match reports that are extrapolated.

Parameters **`match`** (`boolean`) – `true` to match any extrapolated reports, `false` to match non-extrapolated reports

compliance: mandatory – This method must be implemented.

`extrapolated_terms`

Osid Capsule Query

class `dlkit.osid.queries.`**`OsidCapsuleQuery`**

Bases: *`dlkit.osid.queries.OsidQuery`*

This is the query interface for searching capsulating interfaces.

Each method specifies an `AND` term while multiple invocations of the same method produce a nested `OR`.

Metadata

Metadata

class `dlkit.osid.metadata.`**`Metadata`**

The `Metadata` interface defines a set of methods describing a the syntax and rules for creating and updating a data element inside an `OsidForm`.

This interface provides a means to retrieve special restrictions placed upon data elements such as sizes and ranges that may vary from provider to provider or from object to object.

`element_id`

Gets a unique `Id` for the data element.

Returns an `Id`

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`element_label`

Gets a display label for the data element.

Returns a display label

Return type `osid.locale.DisplayText`

compliance: mandatory – This method must be implemented.

`instructions`

Gets instructions for updating this element value.

This is a human readable description of the data element or property that may include special instructions or caveats to the end-user above and beyond what this interface provides.

Returns instructions

Return type `osid.locale.DisplayText`

compliance: mandatory – This method must be implemented.

`syntax`

Gets the syntax of this data.

Returns an enumeration indicating thetype of value

Return type `osid.Syntax`

compliance: mandatory – This method must be implemented.

`is_array``()`

Tests if this data element is an array.

Returns `true` if this data is an array, `false` if a single element

Return type `boolean`

compliance: mandatory – This method must be implemented.

`is_required``()`

Tests if this data element is required for creating new objects.

Returns `true` if this element value is required, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

is_read_only()

Tests if this data can be updated.

This may indicate the result of a pre-authorization but is not a guarantee that an authorization failure will not occur when the create or update transaction is issued.

Returns `true` if this data is not updatable, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

is_linked()

Tests if this data element is linked to other data in the object.

Updating linked data elements should refresh all metadata and revalidate object elements.

Returns true if this element is linked, false if updates have no side effect

Return type `boolean`

compliance: mandatory – This method must be implemented.

is_value_known()

Tests if an existing value is known for this data element.

If it is known that a value does not exist, then this method returns `true`.

Returns `true` if the element value is known, `false` if the element value is not known

Return type `boolean`

compliance: mandatory – This method must be implemented.

has_value()

Tests if this data element has a set non-default value.

Returns `true` if this element value has been set, `false` otherwise

Return type `boolean`

Raise `IllegalState` – `is_value_known()` is `false`

compliance: mandatory – This method must be implemented.

units

Gets the units of this data for display purposes ('lbs', 'gills', 'furlongs').

Returns the display units of this data or an empty string if not applicable

Return type `osid.locale.DisplayText`

compliance: mandatory – This method must be implemented.

minimum_elements

In the case where an array or list of elements is specified in an `OsidForm`, this specifies the minimum number of elements that must be included.

Returns the minimum elements or `1` if `is_array()` is `false`

Return type `cardinal`

compliance: mandatory – This method must be implemented.

maximum_elements

In the case where an array or list of elements is specified in an `OsidForm`, this specifies the maximum number of elements that can be specified.

Returns the maximum elements or `1` if `is_array()` is `false`

Return type `cardinal`

compliance: mandatory – This method must be implemented.

`minimum_cardinal`

Gets the minimum cardinal value.

Returns the minimum cardinal

Return type `cardinal`

Raise `IllegalState` – syntax is not a `CARDINAL`

compliance: mandatory – This method must be implemented.

`maximum_cardinal`

Gets the maximum cardinal value.

Returns the maximum cardinal

Return type `cardinal`

Raise `IllegalState` – syntax is not a `CARDINAL`

compliance: mandatory – This method must be implemented.

`cardinal_set`

Gets the set of acceptable cardinal values.

Returns a set of cardinals or an empty array if not restricted

Return type `cardinal`

Raise `IllegalState` – syntax is not a `CARDINAL`

compliance: mandatory – This method must be implemented.

`default_cardinal_values`

Gets the default cardinal values.

These are the values used if the element value is not provided or is cleared. If `is_array()` is false, then this method returns at most a single value.

Returns the default cardinal values

Return type `cardinal`

Raise `IllegalState` – syntax is not a `CARDINAL` or `is_required()` is `true`

compliance: mandatory – This method must be implemented.

`existing_cardinal_values`

Gets the existing cardinal values.

If `has_value()` and `is_required()` are `false,` then these values are the default values “`. If ``is_array()` is false, then this method returns at most a single value.

Returns the existing cardinal values

Return type `cardinal`

Raise `IllegalState` – syntax is not a `CARDINAL` or `is_value_known()` is `false`

compliance: mandatory – This method must be implemented.

`coordinate_types`

Gets the set of acceptable coordinate types.

Returns the set of coordinate types

Return type `osid.type.Type`

Raise `IllegalState` – syntax is not a `COORDINATE or SPATIALUNIT`

compliance: mandatory – This method must be implemented.

`supports_coordinate_type`(*coordinate_type*)
Tests if the given coordinate type is supported.

Parameters **`coordinate_type`** (`osid.type.Type`) – a coordinate Type

Returns `true` if the type is supported, `false` otherwise

Return type `boolean`

Raise `IllegalState` – syntax is not a `COORDINATE`

Raise `NullArgument` – `coordinate_type` is `null`

compliance: mandatory – This method must be implemented.

`get_axes_for_coordinate_type`(*coordinate_type*)
Gets the number of axes for a given supported coordinate type.

Parameters **`coordinate_type`** (`osid.type.Type`) – a coordinate Type

Returns the number of axes

Return type `cardinal`

Raise `IllegalState` – syntax is not a `COORDINATE`

Raise `NullArgument` – `coordinate_type` is `null`

Raise `Unsupported` – `supports_coordinate_type(coordinate_type)` is `false`

compliance: mandatory – This method must be implemented.

`get_minimum_coordinate_values`(*coordinate_type*)
Gets the minimum coordinate values given supported coordinate type.

Parameters **`coordinate_type`** (`osid.type.Type`) – a coordinate Type

Returns the minimum coordinate values

Return type `decimal`

Raise `IllegalState` – syntax is not a `COORDINATE`

Raise `NullArgument` – `coordinate_type` is `null`

Raise `Unsupported` – `supports_coordinate_type(coordinate_type)` is `false`

compliance: mandatory – This method must be implemented.

`get_maximum_coordinate_values`(*coordinate_type*)
Gets the maximum coordinate values given supported coordinate type.

Parameters **`coordinate_type`** (`osid.type.Type`) – a coordinate Type

Returns the maximum coordinate values

Return type `decimal`

Raise `IllegalState` – syntax is not a `COORDINATE`

Raise `NullArgument` – `coordinate_type` is `null`

Raise `Unsupported` – `supports_coordinate_type(coordinate_type)` is `false`

compliance: mandatory – This method must be implemented.

coordinate_set

Gets the set of acceptable coordinate values.

Returns a set of coordinates or an empty array if not restricted

Return type `osid.mapping.Coordinate`

Raise `IllegalState` – syntax is not a `COORDINATE`

compliance: mandatory – This method must be implemented.

default_coordinate_values

Gets the default coordinate values.

These are the values used if the element value is not provided or is cleared. If `is_array()` is false, then this method returns at most a single value.

Returns the default coordinate values

Return type `osid.mapping.Coordinate`

Raise `IllegalState` – syntax is not a `COORDINATE` or `is_required()` is `true`

compliance: mandatory – This method must be implemented.

existing_coordinate_values

Gets the existing coordinate values.

If `has_value()` and `is_required()` are `false`, then these values are the default values “`. If ``is_array()` is false, then this method returns at most a single value.

Returns the existing coordinate values

Return type `osid.mapping.Coordinate`

Raise `IllegalState` – syntax is not a `COORDINATE` or `is_value_known()` is `false`

compliance: mandatory – This method must be implemented.

currency_types

Gets the set of acceptable currency types.

Returns the set of currency types

Return type `osid.type.Type`

Raise `IllegalState` – syntax is not a `CURRENCY`

compliance: mandatory – This method must be implemented.

supports_currency_type(*currency_type*)

Tests if the given currency type is supported.

Parameters **currency_type** (`osid.type.Type`) – a currency Type

Returns `true` if the type is supported, `false` otherwise

Return type `boolean`

Raise `IllegalState` – syntax is not a `CURRENCY`

Raise `NullArgument` – `currency_type` is `null`

compliance: mandatory – This method must be implemented.

minimum_currency

Gets the minimum currency value.

Returns the minimum currency

Return type `osid.financials.Currency`

Raise `IllegalState` – syntax is not a `CURRENCY`

compliance: mandatory – This method must be implemented.

maximum_currency

Gets the maximum currency value.

Returns the maximum currency

Return type `osid.financials.Currency`

Raise `IllegalState` – syntax is not a `CURRENCY`

compliance: mandatory – This method must be implemented.

currency_set

Gets the set of acceptable currency values.

Returns a set of currencies or an empty array if not restricted

Return type `osid.financials.Currency`

Raise `IllegalState` – syntax is not a `CURRENCY`

compliance: mandatory – This method must be implemented.

default_currency_values

Gets the default currency values.

These are the values used if the element value is not provided or is cleared. If `is_array()` is false, then this method returns at most a single value.

Returns the default currency values

Return type `osid.financials.Currency`

Raise `IllegalState` – syntax is not a `CURRENCY` or `is_required()` is `true`

compliance: mandatory – This method must be implemented.

existing_currency_values

Gets the existing currency values.

If `has_value()` and `is_required()` are `false, then these values are the default values “. If ``is_array()` is false, then this method returns at most a single value.

Returns the existing currency values

Return type `osid.financials.Currency`

Raise `IllegalState` – syntax is not a `CURRENCY` or `is_value_known()` is `false`

compliance: mandatory – This method must be implemented.

date_time_resolution

Gets the smallest resolution of the date time value.

Returns the resolution

Return type `osid.calendaring.DateTimeResolution`

Raise `IllegalState` – syntax is not a `DATETIME, DURATION` , or `TIME`

compliance: mandatory – This method must be implemented.

`calendar_types`

Gets the set of acceptable calendar types.

Returns the set of calendar types

Return type `osid.type.Type`

Raise `IllegalState` – syntax is not a `DATETIME` or `DURATION`

compliance: mandatory – This method must be implemented.

`supports_calendar_type`(*calendar_type*)

Tests if the given calendar type is supported.

Parameters **`calendar_type`** (`osid.type.Type`) – a calendar Type

Returns `true` if the type is supported, `false` otherwise

Return type `boolean`

Raise `IllegalState` – syntax is not a `DATETIME` or `DURATION`

Raise `NullArgument` – `calendar_type` is `null`

compliance: mandatory – This method must be implemented.

`time_types`

Gets the set of acceptable time types.

Returns a set of time types or an empty array if not restricted

Return type `osid.type.Type`

Raise `IllegalState` – syntax is not a `DATETIME, DURATION,` or `TIME`

compliance: mandatory – This method must be implemented.

`supports_time_type`(*time_type*)

Tests if the given time type is supported.

Parameters **`time_type`** (`osid.type.Type`) – a time Type

Returns `true` if the type is supported, `false` otherwise

Return type `boolean`

Raise `IllegalState` – syntax is not a `DATETIME, DURATION,` or `TIME`

Raise `NullArgument` – `time_type` is `null`

compliance: mandatory – This method must be implemented.

`minimum_date_time`

Gets the minimum date time value.

Returns the minimum value

Return type `osid.calendaring.DateTime`

Raise `IllegalState` – syntax is not a `DATETIME`

compliance: mandatory – This method must be implemented.

`maximum_date_time`

Gets the maximum date time value.

Returns the maximum value

Return type `osid.calendaring.DateTime`

Raise `IllegalState` – syntax is not a `DATETIME`

compliance: mandatory – This method must be implemented.

`date_time_set`

Gets the set of acceptable date time values.

Returns a set of values or an empty array if not restricted

Return type `osid.calendaring.DateTime`

Raise `IllegalState` – syntax is not a `DATETIME`

compliance: mandatory – This method must be implemented.

`default_date_time_values`

Gets the default date time values.

These are the values used if the element value is not provided or is cleared. If `is_array()` is false, then this method returns at most a single value.

Returns the default date time values

Return type `osid.calendaring.DateTime`

Raise `IllegalState` – syntax is not a `DATETIME` or `is_required()` is `true`

compliance: mandatory – This method must be implemented.

`existing_date_time_values`

Gets the existing date time values.

If `has_value()` and `is_required()` are `false,` then these values are the default values “`. If ``is_array()` is false, then this method returns at most a single value.

Returns the existing date time values

Return type `osid.calendaring.DateTime`

Raise `IllegalState` – syntax is not a `DATETIME` or `is_value_known()` is `false`

compliance: mandatory – This method must be implemented.

`decimal_scale`

Gets the number of digits to the right of the decimal point.

Returns the scale

Return type `cardinal`

Raise `IllegalState` – syntax is not a `DECIMAL`

compliance: mandatory – This method must be implemented.

`minimum_decimal`

Gets the minimum decimal value.

Returns the minimum decimal

Return type `decimal`

Raise `IllegalState` – syntax is not a `DECIMAL`

compliance: mandatory – This method must be implemented.

maximum_decimal
Gets the maximum decimal value.

Returns the maximum decimal

Return type `decimal`

Raise `IllegalState` – syntax is not a `DECIMAL`

compliance: mandatory – This method must be implemented.

decimal_set
Gets the set of acceptable decimal values.

Returns a set of decimals or an empty array if not restricted

Return type `decimal`

Raise `IllegalState` – syntax is not a `DECIMAL`

compliance: mandatory – This method must be implemented.

default_decimal_values
Gets the default decimal values.

These are the values used if the element value is not provided or is cleared. If `is_array()` is false, then this method returns at most a single value.

Returns the default decimal values

Return type `decimal`

Raise `IllegalState` – syntax is not a `DECIMAL` or `is_required()` is `true`

compliance: mandatory – This method must be implemented.

existing_decimal_values
Gets the existing decimal values.

If `has_value()` and `is_required()` are `false,` then these values are the default values “`. If ``is_array()` is false, then this method returns at most a single value.

Returns the existing decimal values

Return type `decimal`

Raise `IllegalState` – syntax is not a `DECIMAL` or `is_value_known()` is `false`

compliance: mandatory – This method must be implemented.

distance_resolution
Gets the smallest resolution of the distance value.

Returns the resolution

Return type `osid.mapping.DistanceResolution`

Raise `IllegalState` – syntax is not a `DISTANCE`

compliance: mandatory – This method must be implemented.

minimum_distance
Gets the minimum distance value.

Returns the minimum value

Return type `osid.mapping.Distance`

Raise `IllegalState` – syntax is not a `DISTANCE`

compliance: mandatory – This method must be implemented.

maximum_distance

Gets the maximum distance value.

Returns the maximum value

Return type `osid.mapping.Distance`

Raise `IllegalState` – syntax is not a `DISTANCE`

compliance: mandatory – This method must be implemented.

distance_set

Gets the set of acceptable distance values.

Returns a set of values or an empty array if not restricted

Return type `osid.mapping.Distance`

Raise `IllegalState` – syntax is not a `DISTANCE`

compliance: mandatory – This method must be implemented.

default_distance_values

Gets the default distance values.

These are the values used if the element value is not provided or is cleared. If `is_array()` is false, then this method returns at most a single value.

Returns the default distance values

Return type `osid.mapping.Distance`

Raise `IllegalState` – syntax is not a `DISTANCE` or `is_required()` is `true`

compliance: mandatory – This method must be implemented.

existing_distance_values

Gets the existing distance values.

If `has_value()` and `is_required()` are `false,` then these values are the default values “`. If ``is_array()` is false, then this method returns at most a single value.

Returns the existing distance values

Return type `osid.mapping.Distance`

Raise `IllegalState` – syntax is not a `DISTANCE` or `is_value_known()` is `false`

compliance: mandatory – This method must be implemented.

minimum_duration

Gets the minimum duration.

Returns the minimum duration

Return type `osid.calendaring.Duration`

Raise `IllegalState` – syntax is not a `DURATION`

compliance: mandatory – This method must be implemented.

maximum_duration

Gets the maximum duration.

Returns the maximum duration

Return type `osid.calendaring.Duration`

Raise `IllegalState` – syntax is not a `DURATION`

compliance: mandatory – This method must be implemented.

duration_set

Gets the set of acceptable duration values.

Returns a set of durations or an empty array if not restricted

Return type `osid.calendaring.Duration`

Raise `IllegalState` – syntax is not a `DURATION`

compliance: mandatory – This method must be implemented.

default_duration_values

Gets the default duration values.

These are the values used if the element value is not provided or is cleared. If `is_array()` is false, then this method returns at most at most a single value.

Returns the default duration values

Return type `osid.calendaring.Duration`

Raise `IllegalState` – syntax is not a DURATION or `is_required()` is `true`

compliance: mandatory – This method must be implemented.

existing_duration_values

Gets the existing duration values.

If `has_value()` and `is_required()` are `false,` then these values are the default values “`. If ``is_array()` is false, then this method returns at most a single value.

Returns the existing duration values

Return type `osid.calendaring.Duration`

Raise `IllegalState` – syntax is not a `DURATION` or `is_value_known()` is `false`

compliance: mandatory – This method must be implemented.

heading_types

Gets the set of acceptable heading types.

Returns a set of heading types or an empty array if not restricted

Return type `osid.type.Type`

Raise `IllegalState` – syntax is not a `HEADING`

compliance: mandatory – This method must be implemented.

supports_heading_type(*heading_type*)

Tests if the given heading type is supported.

Parameters **heading_type** (`osid.type.Type`) – a heading Type

Returns `true` if the type is supported, `false` otherwise

Return type `boolean`

Raise `IllegalState` – syntax is not a `HEADING`

Raise `NullArgument` – `heading_type` is `null`

compliance: mandatory – This method must be implemented.

get_axes_for_heading_type(*heading_type*)
Gets the number of axes for a given supported heading type.

Parameters heading_type (`osid.type.Type`) – a heading Type

Returns the number of axes

Return type `cardinal`

Raise `IllegalState` – syntax is not a `HEADING`

Raise `NullArgument` – `heading_type` is `null`

Raise `Unsupported` – `supports_heading_type(heading_type)` is `false`

compliance: mandatory – This method must be implemented.

get_minimum_heading_values(*heading_type*)
Gets the minimum heading values given supported heading type.

Parameters heading_type (`osid.type.Type`) – a heading Type

Returns the minimum heading values

Return type `decimal`

Raise `IllegalState` – syntax is not a `HEADING`

Raise `NullArgument` – `heading_type` is `null`

Raise `Unsupported` – `supports_heading_type(heading_type)` is `false`

compliance: mandatory – This method must be implemented.

get_maximum_heading_values(*heading_type*)
Gets the maximum heading values given supported heading type.

Parameters heading_type (`osid.type.Type`) – a heading Type

Returns the maximum heading values

Return type `decimal`

Raise `IllegalState` – syntax is not a `HEADING`

Raise `NullArgument` – `heading_type` is `null`

Raise `Unsupported` – `supports_heading_type(heading_type)` is `false`

compliance: mandatory – This method must be implemented.

heading_set
Gets the set of acceptable heading values.

Returns the set of heading

Return type `osid.mapping.Heading`

Raise `IllegalState` – syntax is not a `HEADING`

compliance: mandatory – This method must be implemented.

default_heading_values
Gets the default heading values.

These are the values used if the element value is not provided or is cleared. If `is_array()` is false, then this method returns at most a single value.

Returns the default heading values

Return type `osid.mapping.Heading`

Raise `IllegalState` – syntax is not a `HEADING` or `is_required()` is `true`

compliance: mandatory – This method must be implemented.

`existing_heading_values`

Gets the existing heading values.

If `has_value()` and `is_required()` are `false`, then these values are the default values `". If ``is_array()` is false, then this method returns at most a single value.

Returns the existing heading values

Return type `osid.mapping.Heading`

Raise `IllegalState` – syntax is not a `HEADING` or `is_value_known()` is `false`

compliance: mandatory – This method must be implemented.

`id_set`

Gets the set of acceptable `Ids`.

Returns a set of `Ids` or an empty array if not restricted

Return type `osid.id.Id`

Raise `IllegalState` – syntax is not an `ID`

compliance: mandatory – This method must be implemented.

`default_id_values`

Gets the default `Id` values.

These are the values used if the element value is not provided or is cleared. If `is_array()` is false, then this method returns at most a single value.

Returns the default `Id` values

Return type `osid.id.Id`

Raise `IllegalState` – syntax is not an `ID` or `is_required()` is `true`

compliance: mandatory – This method must be implemented.

`existing_id_values`

Gets the existing `Id` values.

If `has_value()` and `is_required()` are `false`, then these values are the default values `". If ``is_array()` is false, then this method returns at most a single value.

Returns the existing `Id` values

Return type `osid.id.Id`

Raise `IllegalState` – syntax is not an `ID`

compliance: mandatory – This method must be implemented.

`minimum_integer`

Gets the minimum integer value.

Returns the minimum value

Return type `integer`

Raise `IllegalState` – syntax is not an `INTEGER`

compliance: mandatory – This method must be implemented.

maximum_integer
Gets the maximum integer value.

Returns the maximum value

Return type `integer`

Raise `IllegalState` – syntax is not an `INTEGER`

compliance: mandatory – This method must be implemented.

integer_set
Gets the set of acceptable integer values.

Returns a set of values or an empty array if not restricted

Return type `integer`

Raise `IllegalState` – syntax is not an `INTEGER`

compliance: mandatory – This method must be implemented.

default_integer_values
Gets the default integer values.

These are the values used if the element value is not provided or is cleared. If `is_array()` is false, then this method returns at most a single value.

Returns the default integer values

Return type `integer`

Raise `IllegalState` – syntax is not an `INTEGER` or `is_required()` is `true`

compliance: mandatory – This method must be implemented.

existing_integer_values
Gets the existing integer values.

If `has_value()` and `is_required()` are `false`, then these values are the default values “`. If ``is_array()` is false, then this method returns at most a single value.

Returns the existing integer values

Return type `integer`

Raise `IllegalState` – syntax is not a `INTEGER` or isValueKnown() is false

compliance: mandatory – This method must be implemented.

object_types
Gets the set of acceptable `Types` for an arbitrary object.

Returns a set of `Types` or an empty array if not restricted

Return type `osid.type.Type`

Raise `IllegalState` – syntax is not an `OBJECT`

compliance: mandatory – This method must be implemented.

supports_object_type(*object_type*)
Tests if the given object type is supported.

Parameters **object_type** (`osid.type.Type`) – an object Type

Returns `true` if the type is supported, `false` otherwise

Return type `boolean`

Raise `IllegalState` – syntax is not an `OBJECT`

Raise `NullArgument` – `object_type` is `null`

compliance: mandatory – This method must be implemented.

`object_set`

Gets the set of acceptable object values.

Returns a set of values or an empty array if not restricted

Return type `object`

Raise `IllegalState` – syntax is not an `OBJECT`

compliance: mandatory – This method must be implemented.

`default_object_values`

Gets the default object values.

These are the values used if the element value is not provided or is cleared. If `is_array()` is false, then this method returns at most a single value.

Returns the default object values

Return type `object`

Raise `IllegalState` – syntax is not an `OBJECT` or `is_required()` is `true`

compliance: mandatory – This method must be implemented.

`existing_object_values`

Gets the existing object values.

If `has_value()` and `is_required()` are `false,` then these values are the default values “`. If ``is_array()` is false, then this method returns at most a single value.

Returns the existing object values

Return type `object`

Raise `IllegalState` – syntax is not an OBJECT or `is_value_known()` is `false`

compliance: mandatory – This method must be implemented.

`spatial_unit_record_types`

Gets the set of acceptable spatial unit record types.

Returns the set of spatial unit types

Return type `osid.type.Type`

Raise `IllegalState` – syntax is not `SPATIALUNIT`

compliance: mandatory – This method must be implemented.

`supports_spatial_unit_record_type`(*spatial_unit_record_type*)

Tests if the given spatial unit record type is supported.

Parameters **`spatial_unit_record_type`** (`osid.type.Type`) – a spatial unit record Type

Returns `true` if the type is supported, `false` otherwise

Return type `boolean`

Raise `IllegalState` – syntax is not an `SPATIALUNIT`

Raise `NullArgument` – `spatial_unit_record_type` is `null`

compliance: mandatory – This method must be implemented.

spatial_unit_set

Gets the set of acceptable spatial unit values.

Returns a set of spatial units or an empty array if not restricted

Return type `osid.mapping.SpatialUnit`

Raise `IllegalState` – syntax is not a `SPATIALUNIT`

compliance: mandatory – This method must be implemented.

default_spatial_unit_values

Gets the default spatial unit values.

These are the values used if the element value is not provided or is cleared. If `is_array()` is false, then this method returns at most a single value.

Returns the default spatial unit values

Return type `osid.mapping.SpatialUnit`

Raise `IllegalState` – syntax is not a `SPATIALUNIT` or `is_required()` is `true`

compliance: mandatory – This method must be implemented.

existing_spatial_unit_values

Gets the existing spatial unit values.

If `has_value()` and `is_required()` are `false,` then these values are the default values “`. If ``is_array()` is false, then this method returns at most a single value.

Returns the existing spatial unit values

Return type `osid.mapping.SpatialUnit`

Raise `IllegalState` – syntax is not a SPATIALUNIT or `is_value_known()` is `false`

compliance: mandatory – This method must be implemented.

minimum_speed

Gets the minimum speed value.

Returns the minimum speed

Return type `osid.mapping.Speed`

Raise `IllegalState` – syntax is not a `SPEED`

compliance: mandatory – This method must be implemented.

maximum_speed

Gets the maximum speed value.

Returns the maximum speed

Return type `osid.mapping.Speed`

Raise `IllegalState` – syntax is not a `SPEED`

compliance: mandatory – This method must be implemented.

speed_set

Gets the set of acceptable speed values.

Returns a set of speeds or an empty array if not restricted

Return type `osid.mapping.Speed`

Raise `IllegalState` – syntax is not a `SPEED`

compliance: mandatory – This method must be implemented.

`default_speed_values`

Gets the default speed values.

These are the values used if the element value is not provided or is cleared. If `is_array()` is false, then this method returns at most a single value.

Returns the default speed values

Return type `osid.mapping.Speed`

Raise `IllegalState` – syntax is not a `SPEED` or `is_required()` is `true`

compliance: mandatory – This method must be implemented.

`existing_speed_values`

Gets the existing speed values.

If `has_value()` and `is_required()` are `false,` then these values are the default values “`. If ``is_array()` is false, then this method returns at most a single value.

Returns the existing speed values

Return type `osid.mapping.Speed`

Raise `IllegalState` – syntax is not a `SPEED` or `is_value_known()` is `false`

compliance: mandatory – This method must be implemented.

`minimum_string_length`

Gets the minimum string length.

Returns the minimum string length

Return type `cardinal`

Raise `IllegalState` – syntax is not a `STRING`

compliance: mandatory – This method must be implemented.

`maximum_string_length`

Gets the maximum string length.

Returns the maximum string length

Return type `cardinal`

Raise `IllegalState` – syntax is not a `STRING`

compliance: mandatory – This method must be implemented.

`string_match_types`

Gets the set of valid string match types for use in validating a string.

If the string match type indicates a regular expression then `get_string_expression()` returns a regular expression.

Returns the set of string match types

Return type `osid.type.Type`

Raise `IllegalState` – syntax is not a `STRING`

compliance: mandatory – This method must be implemented.

supports_string_match_type (*string_match_type*)

Tests if the given string match type is supported.

Parameters **string_match_type** (`osid.type.Type`) – a string match type

Returns `true` if the given string match type Is supported, `false` otherwise

Return type `boolean`

Raise `IllegalState` – syntax is not a `STRING`

Raise `NullArgument` – `string_match_type` is `null`

compliance: mandatory – This method must be implemented.

get_string_expression (*string_match_type*)

Gets the regular expression of an acceptable string for the given string match type.

Parameters **string_match_type** (`osid.type.Type`) – a string match type

Returns the regular expression

Return type `string`

Raise `NullArgument` – `string_match_type` is `null`

Raise `IllegalState` – syntax is not a `STRING`

Raise `Unsupported` – `supports_string_match_type(string_match_type )` is `false`

compliance: mandatory – This method must be implemented.

string_format_types

Gets the set of valid string formats.

Returns the set of valid text format types

Return type `osid.type.Type`

Raise `IllegalState` – syntax is not a `STRING`

compliance: mandatory – This method must be implemented.

string_set

Gets the set of acceptable string values.

Returns a set of strings or an empty array if not restricted

Return type `string`

Raise `IllegalState` – syntax is not a `STRING`

compliance: mandatory – This method must be implemented.

default_string_values

Gets the default string values.

These are the values used if the element value is not provided or is cleared. If `is_array()` is false, then this method returns at most a single value.

Returns the default string values

Return type `string`

Raise `IllegalState` – syntax is not a `STRING` or `is_required()` is `true`

compliance: mandatory – This method must be implemented.

`existing_string_values`
Gets the existing string values.

If `has_value()` and `is_required()` are `false`, then these values are the default values “. `If ``is_array()` is false, then this method returns at most a single value.

Returns the existing string values

Return type `string`

Raise `IllegalState` – syntax is not a `STRING` or `is_value_known()` is `false`

compliance: mandatory – This method must be implemented.

`minimum_time`
Gets the minimum time value.

Returns the minimum time

Return type `osid.calendaring.Time`

Raise `IllegalState` – syntax is not a `TIME`

compliance: mandatory – This method must be implemented.

`maximum_time`
Gets the maximum time value.

Returns the maximum time

Return type `osid.calendaring.Time`

Raise `IllegalState` – syntax is not a `TIME`

compliance: mandatory – This method must be implemented.

`time_set`
Gets the set of acceptable time values.

Returns a set of times or an empty array if not restricted

Return type `osid.calendaring.Time`

Raise `IllegalState` – syntax is not a `TIME`

compliance: mandatory – This method must be implemented.

`default_time_values`
Gets the default time values.

These are the values used if the element value is not provided or is cleared. If `is_array()` is false, then this method returns at most a single value.

Returns the default time values

Return type `osid.calendaring.Time`

Raise `IllegalState` – syntax is not a `TIME` or `is_required()` is `true`

compliance: mandatory – This method must be implemented.

`existing_time_values`
Gets the existing time values.

If `has_value()` and `is_required()` are `false`, then these values are the default values “. `If ``is_array()` is false, then this method returns at most a single value.

Returns the existing time values

Return type `osid.calendaring.Time`

Raise `IllegalState` – syntax is not a `TIME` or `is_value_known()` is `false`

compliance: mandatory – This method must be implemented.

`type_set`
Gets the set of acceptable `Types`.

Returns a set of `Types` or an empty array if not restricted

Return type `osid.type.Type`

Raise `IllegalState` – syntax is not a `TYPE`

compliance: mandatory – This method must be implemented.

`default_type_values`
Gets the default type values.

These are the values used if the element value is not provided or is cleared. If `is_array()` is false, then this method returns at most a single value.

Returns the default type values

Return type `osid.type.Type`

Raise `IllegalState` – syntax is not a `TYPE` or `is_required()` is `true`

compliance: mandatory – This method must be implemented.

`existing_type_values`
Gets the existing type values.

If `has_value()` and `is_required()` are `false,` then these values are the default values “`. If ``is_array()` is false, then this method returns at most a single value.

Returns the existing type values

Return type `osid.type.Type`

Raise `IllegalState` – syntax is not a `TYPE` or `is_value_known()` is `false`

compliance: mandatory – This method must be implemented.

`version_types`
Gets the set of acceptable version types.

Returns the set of version types

Return type `osid.type.Type`

Raise `IllegalState` – syntax is not a `VERSION`

compliance: mandatory – This method must be implemented.

`supports_version_type`(*version_type*)
Tests if the given version type is supported.

Parameters **`version_type`** (`osid.type.Type`) – a version Type

Returns `true` if the type is supported, `false` otherwise

Return type `boolean`

Raise `IllegalState` – syntax is not a `VERSION`

Raise `NullArgument` – `version_type` is `null`

compliance: mandatory – This method must be implemented.

minimum_version
Gets the minumim acceptable `Version`.

Returns the minumim `Version`

Return type `osid.installation.Version`

Raise `IllegalState` – syntax is not a `VERSION`

compliance: mandatory – This method must be implemented.

maximum_version
Gets the maximum acceptable `Version`.

Returns the maximum `Version`

Return type `osid.installation.Version`

Raise `IllegalState` – syntax is not a `VERSION`

compliance: mandatory – This method must be implemented.

version_set
Gets the set of acceptable `Versions`.

Returns a set of `Versions` or an empty array if not restricted

Return type `osid.installation.Version`

Raise `IllegalState` – syntax is not a `VERSION`

compliance: mandatory – This method must be implemented.

default_version_values
Gets the default version values.

These are the values used if the element value is not provided or is cleared. If `is_array()` is false, then this method returns at most a single value.

Returns the default version values

Return type `osid.installation.Version`

Raise `IllegalState` – syntax is not a TIME or isValueKnown() is false

compliance: mandatory – This method must be implemented.

existing_version_values
Gets the existing version values.

If `has_value()` and `is_required()` are `false,` then these values are the default values “`. If ``is_array()` is false, then this method returns at most a single value.

Returns the existing version values

Return type `osid.installation.Version`

Raise `IllegalState` – syntax is not a `VERSION` or `is_value_known()` is `false`

compliance: mandatory – This method must be implemented.

Records

Osid Record

class `dlkit.osid.records.`**`OsidRecord`**

`OsidRecord` is a top-level interface for all record objects.

A record is an auxiliary interface that can be retrieved from an OSID object, query, form or search order that contains method definitions outside the core OSID specification. An OSID record interface specification is identified with a `Type`.

`implements_record_type`(*record_type*)

Tests if the given type is implemented by this record.

Other types than that directly indicated by `get_type()` may be supported through an inheritance scheme where the given type specifies a record that is a parent interface of the interface specified by `getType()`.

Parameters **`record_type`** (`osid.type.Type`) – a type

Returns `true` if the given record `Type` is implemented by this record, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `record_type` is `null`

compliance: mandatory – This method must be implemented.

Rules

Osid Condition

class `dlkit.osid.rules.`**`OsidCondition`**

Bases: *`dlkit.osid.markers.Extensible`*, *`dlkit.osid.markers.Suppliable`*

The `OsidCondition` is used to input conditions into a rule for evaluation.

Osid Input

class `dlkit.osid.rules.`**`OsidInput`**

Bases: *`dlkit.osid.markers.Extensible`*, *`dlkit.osid.markers.Suppliable`*

The `OsidInput` is used to input conditions into a rule for processing.

Osid Result

class `dlkit.osid.rules.`**`OsidResult`**

Bases: *`dlkit.osid.markers.Extensible`*, *`dlkit.osid.markers.Browsable`*

The `OsidResult` is used to retrieve the result of processing a rule.

Proxy

Summary

Proxy Open Service Interface Definitions proxy version 3.0.0

The Proxy OSID helps a consumer map external data, such as that received via a server request, into a Proxy that can be used with OSID proxy managers. Then purpose of this OSID is to modularize assumptions made about the input data into another OSID Provider, such as the authentication or localization information.

The `Proxy` represents the `OsidResult` of an evaluation of the input `OsidCondition` performed by the Proxy OSID Provider. The resulting Proxy is meant to be passed to `OsidProxyManagers`. The Proxy OSID is the glue between the application server environment and the OSID services.

The input data may be anything acceptable to a `ProxyCondition` record Type. The `ProxyCondition` record `Types` are aligned with the application server environment while the `Proxy` record `Types` are aligned with OSID Providers. This alignment poses various interoperability issues and as such it might be helpful to be very broad in what may be specified in a `ProxyCondition` so that this service may produce the variety of `Proxy` records needed by the services in the OSID environment.

Some data is defined in the `ProxyCondition`. This in no way implies support of this input by an OSID Provider. The resulting `OsidSession` indicates what actually happened.

Example

An example using a specifier record for an http request: ProxyCondition condition = proxySession.getProxyCondition(); HttpRequestRecord record = condition.getProxyConditionRecord(httpRequestRecordType); record.setHttpRequest(servletRequest);

Proxy proxy = proxySession.getProxy(condition);

Proxy Open Service Interface Definitions proxy version 3.0.0

The Proxy OSID helps a consumer map external data, such as that received via a server request, into a Proxy that can be used with OSID proxy managers. Then purpose of this OSID is to modularize assumptions made about the input data into another OSID Provider, such as the authentication or localization information.

The `Proxy` represents the `OsidResult` of an evaluation of the input `OsidCondition` performed by the Proxy OSID Provider. The resulting Proxy is meant to be passed to `OsidProxyManagers`. The Proxy OSID is the glue between the application server environment and the OSID services.

The input data may be anything acceptable to a `ProxyCondition` record Type. The `ProxyCondition` record `Types` are aligned with the application server environment while the `Proxy` record `Types` are aligned with OSID Providers. This alignment poses various interoperability issues and as such it might be helpful to be very broad in what may be specified in a `ProxyCondition` so that this service may produce the variety of `Proxy` records needed by the services in the OSID environment.

Some data is defined in the `ProxyCondition`. This in no way implies support of this input by an OSID Provider. The resulting `OsidSession` indicates what actually happened.

Example

An example using a specifier record for an http request: ProxyCondition condition = proxySession.getProxyCondition(); HttpRequestRecord record = condition.getProxyConditionRecord(httpRequestRecordType); record.setHttpRequest(servletRequest);

Proxy proxy = proxySession.getProxy(condition);

Service Managers

Proxy Profile

class `dlkit.services.proxy.`**`ProxyProfile`**

Bases: `dlkit.osid.managers.OsidProfile`

The `ProxyProfile` describes the interoperability among proxy services.

supports_proxy()
Tests if a proxy session is supported.

Returns `true` if proxy is supported `,` `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

proxy_record_types
Gets the supported `Proxy` record interface types.

Returns a list containing the supported `Proxy` record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

proxy_condition_record_types
Gets the supported `ProxyCondition` record interface types.

Returns a list containing the supported `ProxyCondition` record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

Records

Proxy Record

class `dlkit.proxy.records.`**ProxyRecord**
Bases: *dlkit.osid.records.OsidRecord*

A record for a `Proxy`.

The methods specified by the record type are available through the underlying object.

Proxy Condition Record

class `dlkit.proxy.records.`**ProxyConditionRecord**
Bases: *dlkit.osid.records.OsidRecord*

A record for a `ProxyCondition`.

The methods specified by the record type are available through the underlying object.

Rules

Proxy

class `dlkit.proxy.rules.`**Proxy**
Bases: *dlkit.osid.rules.OsidResult*

A `Proxy` is used to transfer external information from an application server into an OSID Provider.

has_authentication()
Tests if an authentication is available.

Returns `true` if an `Authentication` is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

authentication

Gets the `Authentication` for this proxy.

Returns the authentication

Return type `osid.authentication.process.Authentication`

Raise `IllegalState` – `has_authentication()` is `false`

compliance: mandatory – This method must be implemented.

has_effective_agent()

Tests if an effective agent is available.

Returns `true` if an effective agent is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

effective_agent_id

Gets the effective `Agent Id` for this proxy.

Returns the effective agent `Id`

Return type `osid.id.Id`

Raise `IllegalState` – `has_effective_agent()` is `false`

compliance: mandatory – This method must be implemented.

effective_agent

Gets the effective `Agent` for this proxy.

Returns the effective agent

Return type `osid.authentication.Agent`

Raise `IllegalState` – `has_effective_agent()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

has_effective_date()

Tests if an effective date is available.

Returns `true` if an effective date is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

effective_date

Gets the effective date.

Returns the effective date

Return type `timestamp`

Raise `IllegalState` – `has_effective_date()` is `false`

compliance: mandatory – This method must be implemented.

effective_clock_rate

Gets the rate of the clock.

Returns the rate

Return type `decimal`

Raise `IllegalState` – `has_effective_date()` is `false`

compliance: mandatory – This method must be implemented.

locale

Gets the locale.

Returns a locale

Return type `osid.locale.Locale`

compliance: mandatory – This method must be implemented.

has_format_type()

Tests if a `DisplayText` format `Type` is available.

Returns `true` if a format type is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

format_type

Gets the `DisplayText` format `Type`.

Returns the format `Type`

Return type `osid.type.Type`

Raise `IllegalState` – `has_format_type()` is `false`

compliance: mandatory – This method must be implemented.

get_proxy_record(*proxy_record_type*)

Gets the proxy record corresponding to the given `Proxy` record `Type`.

This method is used to retrieve an object implementing the requested record. The `proxy_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(proxy_record_type)` is `true` .

Parameters **proxy_record_type** (`osid.type.Type`) – the type of proxy record to retrieve

Returns the proxy record

Return type `osid.proxy.records.ProxyRecord`

Raise `NullArgument` – `proxy_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(proxy_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Proxy Condition

class `dlkit.proxy.rules.`**`ProxyCondition`**

Bases: *`dlkit.osid.rules.OsidCondition`*

A `ProxyCondition` is used to transfer external information into a proxy.

`effective_agent_id`

`set_effective_date`(*date*, *rate*)

Sets the effective date.

Parameters

- **`date`** (`timestamp`) – a date
- **`rate`** (`decimal`) – the rate at which the clock should tick from the given effective date. 0 is a clock that is fixed

Raise `NullArgument` – `date` is `null`

compliance: mandatory – This method must be implemented.

`language_type`

`script_type`

`calendar_type`

`time_type`

`currency_type`

`unit_system_type`

`format_type`

`get_proxy_condition_record`(*proxy_condition_type*)

Gets the proxy condition record corresponding to the given `Proxy` record `Type`.

This method is used to retrieve an object implementing the requested record. The `proxy_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(proxy_record_type)` is `true` .

Parameters **`proxy_condition_type`** (`osid.type.Type`) – the type of proxy condition record to retrieve

Returns the proxy condition record

Return type `osid.proxy.records.ProxyConditionRecord`

Raise `NullArgument` – `proxy_condition_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(proxy_condition_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Relationship

Summary

Relationship Open Service Interface Definitions relationship version 3.0.0

The Relationship OSID provides the ability to relate and manage data between `OsidObjects`.

Relationships

The Relationship OSID defines a `Relationship` that can be used to explicitly identify a relationship between two OSID `Ids` and manage information specific to the relationship.

The Relationship OSID is a building block on which relationships defined in the context of other OSIDs can be built. Examples of relationships include the enrollment record of a student in a `Course` or the commitment or a person to an `Event`.

The Relationship OSID depends on the relationship Type to indicate the nature of the relationship including its natural ordering between the source and destination `Ids`. A relationship of type "friend" may place the peers in either order and be queryable in either order. A relationship of type "parent" is between a father peer and a son peer, but not the other way around. Queries of the son peer based on the "parent" type is not equiavelent to queries of the father peer based on the "parent" type.

Such directional relationships may be accompanied by two types. An additional relationship type of "child" can be used with the son peer to determine the father peer. The directionality and the inverse among the types are part of the type definition.

Family Cataloging

`Relationships` may be cataloged using the `Family` interface.

Sub Packages

The Relationship OSID includes a Relationship Rules OSID for controlling the enable status of `Relationships`.
Relationship Open Service Interface Definitions relationship version 3.0.0

The Relationship OSID provides the ability to relate and manage data between `OsidObjects`.

Relationships

The Relationship OSID defines a `Relationship` that can be used to explicitly identify a relationship between two OSID `Ids` and manage information specific to the relationship.

The Relationship OSID is a building block on which relationships defined in the context of other OSIDs can be built. Examples of relationships include the enrollment record of a student in a `Course` or the commitment or a person to an `Event`.

The Relationship OSID depends on the relationship Type to indicate the nature of the relationship including its natural ordering between the source and destination `Ids`. A relationship of type "friend" may place the peers in either order and be queryable in either order. A relationship of type "parent" is between a father peer and a son peer, but not the other way around. Queries of the son peer based on the "parent" type is not equiavelent to queries of the father peer based on the "parent" type.

Such directional relationships may be accompanied by two types. An additional relationship type of "child" can be used with the son peer to determine the father peer. The directionality and the inverse among the types are part of the type definition.

Family Cataloging

`Relationships` may be cataloged using the `Family` interface.

Sub Packages

The Relationship OSID includes a Relationship Rules OSID for controlling the enable status of `Relationships`.

Service Managers

Relationship Profile

class `dlkit.services.relationship.`**`RelationshipProfile`**

Bases: `dlkit.osid.managers.OsidProfile`

The relationship profile describes the interoperability among relationship services.

`supports_relationship_lookup``()`

Tests if looking up relationships is supported.

Returns `true` if relationship lookup is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_relationship_query``()`

Tests if querying relationships is supported.

Returns `true` if relationship query is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_relationship_admin``()`

Tests if relationship administrative service is supported.

Returns `true` if relationship administration is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_family_lookup``()`

Tests if looking up families is supported.

Returns `true` if family lookup is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_family_admin``()`

Tests if familyadministrative service is supported.

Returns `true` if family administration is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_family_hierarchy``()`

Tests for the availability of a family hierarchy traversal service.

Returns `true` if family hierarchy traversal is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented in all providers.

supports_family_hierarchy_design()

Tests for the availability of a family hierarchy design service.

Returns `true` if family hierarchy design is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

relationship_record_types

Gets the supported `Relationship` record types.

Returns a list containing the supported `Relationship` record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

relationship_search_record_types

Gets the supported `Relationship` search record types.

Returns a list containing the supported `Relationship` search record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

family_record_types

Gets the supported `Family` record types.

Returns a list containing the supported `Family` types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

family_search_record_types

Gets the supported `Family` search record types.

Returns a list containing the supported `Family` search record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

Relationship Manager

class `dlkit.services.relationship.`**RelationshipManager**(*proxy=None*)

Bases: `dlkit.osid.managers.OsidManager`, `dlkit.osid.sessions.OsidSession`, *`dlkit.services.relationship.RelationshipProfile`*

The relationship manager provides access to relationship sessions and provides interoperability tests for various aspects of this service.

The sessions included in this manager are:

- `RelationshipLookupSession`: a session to retrieve and examine relationships
- `RelationshipQuerySession`: a session to query relationships
- `RelationshipSearchSession`: a session to search for relationships
- `RelationshipAdminSession`: a session to manage relationships
- `RelationshipNotificationSession`: a session to receive notifications pertaining to relationship changes

- `RelationshipFamilySession`: a session to look up relationship to family mappings
- `RelationshipFamilyAssignmentSession`: a session to manage relationship to family catalog mappings
- `RelationshipSmartFamilySession`: a session to manage dynamic relationship families
- `FamilyLookupSession`: a session to retrieve families
- `FamilyQuerySession`: a session to query families
- `FamilySearchSession`: a session to search for families
- `FamilyAdminSession`: a session to create and delete families
- `FamilyNotificationSession`: a session to receive notifications pertaining to family changes
- `FamilyHierarchySession`: a session to traverse a hierarchy of families
- `FamilyHierarchyDesignSession`: a session to manage a family hierarchy

relationship_batch_manager

Gets the relationship batch manager.

Returns a `RelationshipBatchManager`

Return type `osid.relationship.batch.RelationshipBatchManager`

Raise `OperationFailed` – unable to complete request

Raise `Unimplemented` – `supports_relationship_batch()` is `false`

compliance: optional – This method must be implemented if ``supports_relationship_batch()`` is ``true``.

relationship_rules_manager

Gets the relationship rules manager.

Returns a `RelationshipRulesManager`

Return type `osid.relationship.rules.RelationshipRulesManager`

Raise `OperationFailed` – unable to complete request

Raise `Unimplemented` – `supports_relationship_rules()` is `false`

compliance: optional – This method must be implemented if ``supports_relationship_rules()`` is ``true``.

Family Lookup Methods

`RelationshipManager.`**can_lookup_families**`()`

Tests if this user can perform `Family` lookups.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may not offer lookup operations to unauthorized users.

Returns `false` if lookup methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**use_comparative_family_view**`()`

The returns from the family methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`RelationshipManager.`**`use_plenary_family_view`**`()`

A complete view of the `Family` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`RelationshipManager.`**`get_family`**`(`*family_id*`)`

Gets the `Family` specified by its `Id`.

In plenary mode, the exact `Id` is found or a `NotFound` results. Otherwise, the returned `Family` may have a different `Id` than requested, such as the case where a duplicate `Id` was assigned to a `Family` and retained for compatibil

Parameters **`family_id`** (`osid.id.Id`) – `Id` of the `Family`

Returns the family

Return type `osid.relationship.Family`

Raise `NotFound` – `family_id` not found

Raise `NullArgument` – `family_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method is must be implemented.

`RelationshipManager.`**`get_families_by_ids`**`(`*family_ids*`)`

Gets a `FamilyList` corresponding to the given `IdList`.

In plenary mode, the returned list contains all of the families specified in the `Id` list, in the order of the list, including duplicates, or an error results if an `Id` in the supplied list is not found or inaccessible. Otherwise, inaccessible families may be omitted from the list and may present the elements in any order including returning a unique set.

Parameters **`family_ids`** (`osid.id.IdList`) – the list of `Ids` to retrieve

Returns the returned `Family list`

Return type `osid.relationship.FamilyList`

Raise `NotFound` – an `Id was` not found

Raise `NullArgument` – `family_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`get_families_by_genus_type`**`(`*family_genus_type*`)`

Gets a `FamilyList` corresponding to the given family genus `Type` which does not include families of genus types derived from the specified `Type`.

In plenary mode, the returned list contains all known families or an error results. Otherwise, the returned list may contain only those families that are accessible through this session.

Parameters **`family_genus_type`** (`osid.type.Type`) – a family genus type

Returns the returned `Family list`

Return type `osid.relationship.FamilyList`

Raise `NullArgument` – `family_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`get_families_by_parent_genus_type`**(*family_genus_type*)

Gets a `FamilyList` corresponding to the given family genus `Type` and include any additional families with genus types derived from the specified `Type`.

In plenary mode, the returned list contains all known families or an error results. Otherwise, the returned list may contain only those families that are accessible through this session.

Parameters **`family_genus_type`** (`osid.type.Type`) – a family genus type

Returns the returned `Family list`

Return type `osid.relationship.FamilyList`

Raise `NullArgument` – `family_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`get_families_by_record_type`**(*family_record_type*)

Gets a `FamilyList` containing the given family record `Type`.

In plenary mode, the returned list contains all known families or an error results. Otherwise, the returned list may contain only those families that are accessible through this session.

Parameters **`family_record_type`** (`osid.type.Type`) – a family record type

Returns the returned `Family list`

Return type `osid.relationship.FamilyList`

Raise `NullArgument` – `family_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`get_families_by_provider`**(*resource_id*)

Gets a `FamilyList` from the given provider.

In plenary mode, the returned list contains all known families or an error results. Otherwise, the returned list may contain only those families that are accessible through this session.

Parameters **`resource_id`** (`osid.id.Id`) – a resource `Id`

Returns the returned `Family list`

Return type `osid.relationship.FamilyList`

Raise `NullArgument` – `resource_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`families`**

Gets all families.

In plenary mode, the returned list contains all known families or an error results. Otherwise, the returned list may contain only those families that are accessible through this session.

Returns a list of families

Return type `osid.relationship.FamilyList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Family Admin Methods

`RelationshipManager.`**`can_create_families`**`()`

Tests if this user can create families.

A return of true does not guarantee successful authorization. A return of false indicates that it is known creating a `Family` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer create operations to unauthorized users.

Returns `false` if `Family` creation is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`can_create_family_with_record_types`**(*family_record_types*)

Tests if this user can create a single `Family` using the desired record types.

While `RelationshipManager.getFamilyRecordTypes()` can be used to examine which records are supported, this method tests which record(s) are required for creating a specific `Family`. Providing an empty array tests if a `Family` can be created with no records.

Parameters **`family_record_types`** (`osid.type.Type[]`) – array of family record types

Returns `true` if `Family` creation using the specified record `Types` is supported, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `family_record_types is null`

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`get_family_form_for_create`**(*family_record_types*)

Gets the family form for creating new families.

A new form should be requested for each create transaction.

Parameters **`family_record_types`** (`osid.type.Type[]`) – array of family record types

Returns the family form

Return type `osid.relationship.FamilyForm`

Raise `NullArgument` – `family_record_types is null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – unable to get form for requested record types

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`create_family`**`(`*family_form*`)`

Creates a new `Family`.

Parameters **`family_form`** (`osid.relationship.FamilyForm`) – the form for this `Family`.

Returns the new `Family`

Return type `osid.relationship.Family`

Raise `IllegalState` – `family_form` already used in a create transaction

Raise `InvalidArgument` – one or more of the form elements is invalid

Raise `NullArgument` – `family_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `family_form` did not originate from `get_family_form_for_create()`

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`can_update_families`**`()`

Tests if this user can update families.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating a `Family` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer update operations to unauthorized users.

Returns `false` if `Family` modification is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`get_family_form_for_update`**`(`*family_id*`)`

Gets the family form for updating an existing family.

A new family form should be requested for each update transaction.

Parameters **`family_id`** (`osid.id.Id`) – the `Id` of the `Family`

Returns the family form

Return type `osid.relationship.FamilyForm`

Raise `NotFound` – `family_id` is not found

Raise `NullArgument` – `family_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`update_family`**`(`*family_form*`)`

Updates an existing family.

Parameters **family_form** (osid.relationship.FamilyForm) – the form containing the elements to be updated

Raise IllegalState – family_form already used in an update transaction

Raise InvalidArgument – the form contains an invalid value

Raise NullArgument – family_id or family_form is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

Raise Unsupported – family_form did not originate from get_family_form_for_update()

compliance: mandatory – This method must be implemented.

RelationshipManager.**can_delete_families**()

Tests if this user can delete families.

A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting a Family will result in a PermissionDenied. This is intended as a hint to an application that may not wish to offer delete operations to unauthorized users.

Returns false if Family deletion is not authorized, true otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

RelationshipManager.**delete_family**(*family_id*)

Deletes a Family.

Parameters **family_id** (osid.id.Id) – the Id of the Family to remove

Raise NotFound – family_id not found

Raise NullArgument – family_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

RelationshipManager.**can_manage_family_aliases**()

Tests if this user can manage Id aliases for families.

A return of true does not guarantee successful authorization. A return of false indicates that it is known changing an alias will result in a PermissionDenied. This is intended as a hint to an application that may opt not to offer alias operations to an unauthorized user.

Returns false if Family aliasing is not authorized, true otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

RelationshipManager.**alias_family**(*family_id*, *alias_id*)

Adds an Id to a Family for the purpose of creating compatibility.

The primary Id of the Family is determined by the provider. The new Id performs as an alias to the primary Id. If the alias is a pointer to another family, it is reassigned to the given family Id.

Parameters

- **family_id** (osid.id.Id) – the Id of a Family

- **alias_id** (`osid.id.Id`) – the alias `Id`

Raise `AlreadyExists` – `alias_id` is already assigned

Raise `NotFound` – `family_id` not found

Raise `NullArgument` – `family_id` or `alias_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Family Hierarchy Methods

`RelationshipManager.`**`family_hierarchy_id`**

Gets the hierarchy `Id` associated with this session.

Returns the hierarchy `Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`family_hierarchy`**

Gets the hierarchy associated with this session.

Returns the hierarchy associated with this session

Return type `osid.hierarchy.Hierarchy`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`can_access_family_hierarchy`**`()`

Tests if this user can perform hierarchy queries.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an an application that may not offer hierrachy traversal operations to unauthorized users.

Returns `false` if hierarchy traversal methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`use_comparative_family_view`**`()`

The returns from the family methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`RelationshipManager.`**`use_plenary_family_view`**`()`

A complete view of the `Family` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`RelationshipManager.`**`root_family_ids`**

Gets the root family `Ids` in this hierarchy.

Returns the root family `Ids`

Return type `osid.id.IdList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`root_families`**

Gets the root families in the family hierarchy.

A node with no parents is an orphan. While all family `Ids` are known to the hierarchy, an orphan does not appear in the hierarchy unless explicitly added as a root node or child of another node.

Returns the root families

Return type `osid.relationship.FamilyList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method is must be implemented.

`RelationshipManager.`**`has_parent_families`**(*family_id*)

Tests if the `Family` has any parents.

Parameters **`family_id`** (`osid.id.Id`) – the `Id` of a family

Returns `true` if the family has parents, `false` otherwise

Return type `boolean`

Raise `NotFound` – `family_id` is not found

Raise `NullArgument` – `family_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`is_parent_of_family`**(*id_*, *family_id*)

Tests if an `Id` is a direct parent of a family.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`family_id`** (`osid.id.Id`) – the `Id` of a family

Returns `true` if this `id` is a parent of `family_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `family_id` is not found

Raise `NullArgument` – `id` or `family_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` not found return `false`.

`RelationshipManager.`**`get_parent_family_ids`**(*family_id*)

Gets the parent `Ids` of the given family.

Parameters **`family_id`** (`osid.id.Id`) – the `Id` of a family

Returns the parent `Ids` of the family

Return type `osid.id.IdList`

Raise `NotFound` – `family_id` is not found

Raise `NullArgument` – `family_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`get_parent_families`**(*family_id*)

Gets the parent families of the given `id`.

Parameters **`family_id`** (`osid.id.Id`) – the `Id` of the `Family` to query

Returns the parent families of the `id`

Return type `osid.relationship.FamilyList`

Raise `NotFound` – a `Family` identified by `Id is` not found

Raise `NullArgument` – `family_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`is_ancestor_of_family`**(*id_*, *family_id*)

Tests if an `Id` is an ancestor of a family.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`family_id`** (`osid.id.Id`) – the `Id` of a family

Returns `true` if this `id` is an ancestor of `family_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `family_id` is not found

Raise `NullArgument` – `id` or `family_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` not found return `false`.

`RelationshipManager.`**`has_child_families`**(*family_id*)

Tests if a family has any children.

Parameters **`family_id`** (`osid.id.Id`) – the `Id` of a family

Returns `true` if the `family_id` has children, `false` otherwise

Return type `boolean`

Raise `NotFound` – `family_id` is not found

Raise `NullArgument` – `family_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`is_child_of_family`**`(`*id_*, *family_id*`)`

Tests if a family is a direct child of another.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`family_id`** (`osid.id.Id`) – the `Id` of a family

Returns `true` if the `id` is a child of `family_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `family_id` is not found

Raise `NullArgument` – `id` or `family_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` not found return `false`.

`RelationshipManager.`**`get_child_family_ids`**`(`*family_id*`)`

Gets the child `Ids` of the given family.

Parameters **`family_id`** (`osid.id.Id`) – the `Id` to query

Returns the children of the family

Return type `osid.id.IdList`

Raise `NotFound` – `family_id` is not found

Raise `NullArgument` – `family_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`get_child_families`**`(`*family_id*`)`

Gets the child families of the given `id`.

Parameters **`family_id`** (`osid.id.Id`) – the `Id` of the `Family` to query

Returns the child families of the `id`

Return type `osid.relationship.FamilyList`

Raise `NotFound` – a `Family` identified by `Id is` not found

Raise `NullArgument` – `family_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`is_descendant_of_family`**(*id_*, *family_id*)

Tests if an `Id` is a descendant of a family.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`family_id`** (`osid.id.Id`) – the `Id` of a family

Returns `true` if the `id` is a descendant of the `family_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `family_id` is not found

Raise `NullArgument` – `id` or `family_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` is not found return `false`.

`RelationshipManager.`**`get_family_node_ids`**(*family_id*, *ancestor_levels*, *descendant_levels*, *include_siblings*)

Gets a portion of the hierarchy for the given family.

Parameters

- **`family_id`** (`osid.id.Id`) – the `Id` to query
- **`ancestor_levels`** (`cardinal`) – the maximum number of ancestor levels to include. A value of 0 returns no parents in the node.
- **`descendant_levels`** (`cardinal`) – the maximum number of descendant levels to include. A value of 0 returns no children in the node.
- **`include_siblings`** (`boolean`) – `true` to include the siblings of the given node, `false` to omit the siblings

Returns a family node

Return type `osid.hierarchy.Node`

Raise `NotFound` – `family_id` is not found

Raise `NullArgument` – `family_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`get_family_nodes`**(*family_id*, *ancestor_levels*, *descendant_levels*, *include_siblings*)

Gets a portion of the hierarchy for the given family.

Parameters

- **`family_id`** (`osid.id.Id`) – the `Id` to query

- **ancestor_levels** (`cardinal`) – the maximum number of ancestor levels to include. A value of 0 returns no parents in the node.
- **descendant_levels** (`cardinal`) – the maximum number of descendant levels to include. A value of 0 returns no children in the node.
- **include_siblings** (`boolean`) – `true` to include the siblings of the given node, `false` to omit the siblings

Returns a family node

Return type `osid.relationship.FamilyNode`

Raise `NotFound` – `family_id` is not found

Raise `NullArgument` – `family_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Family Hierarchy Design Methods

`RelationshipManager.`**`family_hierarchy_id`**

Gets the hierarchy `Id` associated with this session.

Returns the hierarchy `Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`family_hierarchy`**

Gets the hierarchy associated with this session.

Returns the hierarchy associated with this session

Return type `osid.hierarchy.Hierarchy`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`can_modify_family_hierarchy`**`()`

Tests if this user can change the hierarchy.

A return of true does not guarantee successful authorization. A return of false indicates that it is known performing any update will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer these operations to an unauthorized user.

Returns `false` if changing this hierarchy is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`add_root_family`**`(`*family_id*`)`

Adds a root family.

Parameters **family_id** (`osid.id.Id`) – the `Id` of a family

Raise `AlreadyExists` – `family_id` is already in hierarchy

Raise `NotFound` – `family_id` not found

Raise `NullArgument` – `family_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`remove_root_family`**(*family_id*)

Removes a root family.

Parameters **`family_id`** (`osid.id.Id`) – the `Id` of a family

Raise `NotFound` – `family_id` not a root

Raise `NullArgument` – `family_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`add_child_family`**(*family_id*, *child_id*)

Adds a child to a family.

Parameters

- **`family_id`** (`osid.id.Id`) – the `Id` of a family
- **`child_id`** (`osid.id.Id`) – the `Id` of the new child

Raise `AlreadyExists` – `family_id` is already a parent of `child_id`

Raise `NotFound` – `family_id` or `child_id` not found

Raise `NullArgument` – `family_id` or `child_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`remove_child_family`**(*family_id*, *child_id*)

Removes a child from a family.

Parameters

- **`family_id`** (`osid.id.Id`) – the `Id` of a family
- **`child_id`** (`osid.id.Id`) – the `Id` of the new child

Raise `NotFound` – `family_id` not a parent of `child_id`

Raise `NullArgument` – `family_id` or `child_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RelationshipManager.`**`remove_child_families`**(*family_id*)

Removes all children from a family.

Parameters **`family_id`** (`osid.id.Id`) – the `Id` of a family

Raise `NotFound` – `family_id` not in hierarchy

Raise `NullArgument` – `family_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Family

Family

class `dlkit.services.relationship.`**`Family`**(*provider_manager*, *catalog*, *runtime*, *proxy*, ***kwargs*)

Bases: *`dlkit.osid.objects.OsidCatalog`*, `dlkit.osid.sessions.OsidSession`

A `Family` represents a collection of relationships.

Like all OSID objects, a `Family` is identified by its `Id` and any persisted references should use the `Id`.

`get_family_record`(*family_record_type*)

Gets the famly record corresponding to the given `Family` record `Type`.

This method is used to retrieve an object implementing the requested record. The `family_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(family_record_type)` is `true` .

Parameters **`family_record_type`** (`osid.type.Type`) – the type of family record to retrieve

Returns the family record

Return type `osid.relationship.records.FamilyRecord`

Raise `NullArgument` – `family_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – `has_record_type(family_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Relationship Lookup Methods

`Family.`**`family_id`**

Gets the `Familt Id` associated with this session.

Returns the `Family Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Family.`**`family`**

Gets the `Family` associated with this session.

Returns the family

Return type `osid.relationship.Family`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Family.`**`can_lookup_relationships`**`()`

Tests if this user can perform `Relationship` lookups.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may not offer lookup operations to unauthorized users.

Returns `false` if lookup methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Family.`**`use_comparative_relationship_view`**`()`

The returns from the lookup methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`Family.`**`use_plenary_relationship_view`**`()`

A complete view of the `Relationship` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`Family.`**`use_federated_family_view`**`()`

Federates the view for methods in this session.

A federated view will include relationships in families which are children of this family in the family hierarchy.

compliance: mandatory – This method is must be implemented.

`Family.`**`use_isolated_family_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts retrievals to this family only.

compliance: mandatory – This method is must be implemented.

`Family.`**`use_effective_relationship_view`**`()`

Only relationships whose effective dates are current are returned by methods in this session.

compliance: mandatory – This method is must be implemented.

`Family.`**`use_any_effective_relationship_view`**`()`

All relationships of any effective dates are returned by all methods in this session.

compliance: mandatory – This method is must be implemented.

`Family.`**`get_relationship`**`(`*relationship_id*`)`

Gets the `Relationship` specified by its `Id`.

Parameters **`relationship_id`** (`osid.id.Id`) – the `Id` of the `Relationship` to retrieve

Returns the returned `Relationship`

Return type `osid.relationship.Relationship`

Raise `NotFound` – no `Relationship` found with the given `Id`

Raise `NullArgument` – `relationship_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Family.`**`get_relationships_by_ids`**(*relationship_ids*)

Gets a `RelationshipList` corresponding to the given `IdList`.

Parameters **`relationship_ids`** (`osid.id.IdList`) – the list of `Ids` to retrieve

Returns the returned `Relationship list`

Return type `osid.relationship.RelationshipList`

Raise `NotFound` – an `Id` was not found

Raise `NullArgument` – `relationship_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Family.`**`get_relationships_by_genus_type`**(*relationship_genus_type*)

Gets a `RelationshipList` corresponding to the given relationship genus `Type` which does not include relationships of types derived from the specified `Type`.

Parameters **`relationship_genus_type`** (`osid.type.Type`) – a relationship genus type

Returns the returned `Relationship list`

Return type `osid.relationship.RelationshipList`

Raise `NullArgument` – `relationship_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Family.`**`get_relationships_by_parent_genus_type`**(*relationship_genus_type*)

Gets a `RelationshipList` corresponding to the given relationship genus `Type` and include any additional relationships with genus types derived from the specified `Type`.

Parameters **`relationship_genus_type`** (`osid.type.Type`) – a relationship genus type

Returns the returned `Relationship list`

Return type `osid.relationship.RelationshipList`

Raise `NullArgument` – `relationship_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Family.**get_relationships_by_record_type**(*relationship_record_type*)
Gets a RelationshipList containing the given relationship record Type.

Parameters **relationship_record_type** (osid.type.Type) – a relationship record type

Returns the returned RelationshipList

Return type osid.relationship.RelationshipList

Raise NullArgument – relationship_record_type is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Family.**get_relationships_on_date**(*from_*, *to*)
Gets a RelationshipList effective during the entire given date range inclusive but not confined to the date range.

Parameters

- **from** (osid.calendaring.DateTime) – starting date
- **to** (osid.calendaring.DateTime) – ending date

Returns the relationships

Return type osid.relationship.RelationshipList

Raise InvalidArgument – from is greater than to

Raise NullArgument – from or to is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Family.**get_relationships_for_source**(*source_id*)
Gets a RelationshipList corresponding to the given peer Id.

Parameters **source_id** (osid.id.Id) – a peer Id

Returns the relationships

Return type osid.relationship.RelationshipList

Raise NullArgument – source_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Family.**get_relationships_for_source_on_date**(*source_id*, *from_*, *to*)
Gets a RelationshipList corresponding to the given peer Id and effective during the entire given date range inclusive but not confined to the date range.

Parameters

- **source_id** (osid.id.Id) – a peer Id
- **from** (osid.calendaring.DateTime) – starting date

- **to** (`osid.calendaring.DateTime`) – ending date

Returns the relationships

Return type `osid.relationship.RelationshipList`

Raise `InvalidArgument` – `from is greater than to`

Raise `NullArgument` – `source_id, from` ,or `to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Family.`**`get_relationships_by_genus_type_for_source`**(*source_id*, *relationship_genus_type*)

Gets a `RelationshipList` corresponding to the given peer `Id` and relationship genus ``Type.

Relationships`` of any genus derived from the given genus are returned.

In plenary mode, the returned list contains all of the relationships corresponding to the given peer, including duplicates, or an error results if a relationship is inaccessible. Otherwise, inaccessible `Relationships` may be omitted from the list and may present the elements in any order including returning a unique set.

In effective mode, relationships are returned that are currently effective. In any effective mode, effective relationships and those currently expired are returned.

Parameters

- **source_id** (`osid.id.Id`) – a peer `Id`
- **relationship_genus_type** (`osid.type.Type`) – a relationship genus type

Returns the relationships

Return type `osid.relationship.RelationshipList`

Raise `NullArgument` – `source_id` or `relationship_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Family.`**`get_relationships_by_genus_type_for_source_on_date`**(*source_id*, *relationship_genus_type*, *from_*, *to*)

Gets a `RelationshipList` corresponding to the given peer `Id` and relationship genus `Type` and effective during the entire given date range inclusive but not confined to the date range.

Parameters

- **source_id** (`osid.id.Id`) – a peer `Id`
- **relationship_genus_type** (`osid.type.Type`) – a relationship genus type
- **from** (`osid.calendaring.DateTime`) – starting date
- **to** (`osid.calendaring.DateTime`) – ending date

Returns the relationships

Return type `osid.relationship.RelationshipList`

Raise `InvalidArgument` – `from is greater than to`

Raise `NullArgument` – `source_id, relationship_genus_type, from` or `to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Family.`**`get_relationships_for_destination`**(*destination_id*)

Gets a `RelationshipList` corresponding to the given peer `Id`.

Parameters **`destination_id`** (`osid.id.Id`) – a peer `Id`

Returns the relationships

Return type `osid.relationship.RelationshipList`

Raise `NullArgument` – `destination_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Family.`**`get_relationships_for_destination_on_date`**(*destination_id*, *from_*, *to*)

Gets a `RelationshipList` corresponding to the given peer `Id` with a starting effective date in the given range inclusive.

Parameters

- **`destination_id`** (`osid.id.Id`) – a peer `Id`
- **`from`** (`osid.calendaring.DateTime`) – starting date
- **`to`** (`osid.calendaring.DateTime`) – ending date

Returns the relationships

Return type `osid.relationship.RelationshipList`

Raise `InvalidArgument` – `from is greater than to`

Raise `NullArgument` – `destination_id, from` ,or `to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Family.`**`get_relationships_by_genus_type_for_destination`**(*destination_id*, *relationship_genus_type*)

Gets a `RelationshipList` corresponding to the given peer `Id` and relationship genus ``Type.

Relationships`` of any genus derived from the given genus are returned.

In plenary mode, the returned list contains all of the relationships corresponding to the given peer, including duplicates, or an error results if a relationship is inaccessible. Otherwise, inaccessible `Relationships` may be omitted from the list and may present the elements in any order including returning a unique set.

In effective mode, relationships are returned that are currently effective. In any effective mode, effective relationships and those currently expired are returned.

Parameters

- **destination_id** (`osid.id.Id`) – a peer `Id`
- **relationship_genus_type** (`osid.type.Type`) – a relationship genus type

Returns the relationships

Return type `osid.relationship.RelationshipList`

Raise `NullArgument` – `destination_id` or `relationship_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Family.`**`get_relationships_by_genus_type_for_destination_on_date`**(*destination_id*, *relationship_genus_type*, *from_*, *to*)

Gets a `RelationshipList` corresponding to the given peer `Id` and relationship genus `Type` and effective during the entire given date range inclusive but not confined to the date range.

Parameters

- **destination_id** (`osid.id.Id`) – a peer `Id`
- **relationship_genus_type** (`osid.type.Type`) – a relationship genus type
- **from** (`osid.calendaring.DateTime`) – starting date
- **to** (`osid.calendaring.DateTime`) – ending date

Returns the relationships

Return type `osid.relationship.RelationshipList`

Raise `InvalidArgument` – `from is greater than to`

Raise `NullArgument – destination_id, relationship_genus_type, from` or `to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Family.`**`get_relationships_for_peers`**(*source_id*, *destination_id*)

Gets a `RelationshipList` corresponding to the given peer `Ids`.

Parameters

- **source_id** (`osid.id.Id`) – a peer `Id`
- **destination_id** (`osid.id.Id`) – a related peer `Id`

Returns the relationships

Return type `osid.relationship.RelationshipList`

Raise `NullArgument` – `source_id` or `destination_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Family.`**`get_relationships_for_peers_on_date`**(*source_id*, *destination_id*, *from_*, *to*)

Gets a `RelationshipList` corresponding to the given peer `Ids` and effective during the entire given date range inclusive but not confined to the date range.

Parameters

- **`source_id`** (`osid.id.Id`) – a peer `Id`
- **`destination_id`** (`osid.id.Id`) – a related peer `Id`
- **`from`** (`osid.calendaring.DateTime`) – starting date
- **`to`** (`osid.calendaring.DateTime`) – ending date

Returns the relationships

Return type `osid.relationship.RelationshipList`

Raise `InvalidArgument` – `from` is greater than `to`

Raise `NullArgument` – `source_id, destination_id, from` or `to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Family.`**`get_relationships_by_genus_type_for_peers`**(*source_id*, *destination_id*, *relationship_genus_type*)

Gets a `RelationshipList` corresponding between the given peer `Ids` and relationship genus ‘‘Type.

Relationships‘‘ of any genus derived from the given genus are returned.

In plenary mode, the returned list contains all of the relationships corresponding to the given peer or an error results if a relationship is inaccessible. Otherwise, inaccessible `Relationships` may be omitted from the list.

In effective mode, relationships are returned that are currently effective. In any effective mode, effective relationships and those currently expired are returned.

Parameters

- **`source_id`** (`osid.id.Id`) – a peer `Id`
- **`destination_id`** (`osid.id.Id`) – a related peer `Id`
- **`relationship_genus_type`** (`osid.type.Type`) – a relationship genus type

Returns the relationships

Return type `osid.relationship.RelationshipList`

Raise `NullArgument` – `source_id, destination_id,` or `relationship_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Family.`**`get_relationships_by_genus_type_for_peers_on_date`**(*source_id*, *destination_id*, *relationship_genus_type*, *from_*, *to*)

Gets a `RelationshipList` effective during the entire given date range inclusive but not confined to the date range.

Parameters

- **`source_id`** (`osid.id.Id`) – a peer `Id`
- **`destination_id`** (`osid.id.Id`) – a related peer `Id`
- **`relationship_genus_type`** (`osid.type.Type`) – a relationship genus type
- **`from`** (`osid.calendaring.DateTime`) – starting date
- **`to`** (`osid.calendaring.DateTime`) – ending date

Returns the relationships

Return type `osid.relationship.RelationshipList`

Raise `InvalidArgument` – `from is greater than to`

Raise `NullArgument` – `source_id, destination_id, relationship_genus_type, from` or `to` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Family.`**`relationships`**

Gets all `Relationships`.

Returns a list of `Relationships`

Return type `osid.relationship.RelationshipList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Relationship Query Methods

`Family.`**`family_id`**

Gets the `Familt Id` associated with this session.

Returns the `Family Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Family.`**`family`**

Gets the `Family` associated with this session.

Returns the family

Return type `osid.relationship.Family`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Family.`**`use_federated_family_view`**`()`

Federates the view for methods in this session.

A federated view will include relationships in families which are children of this family in the family hierarchy.

compliance: mandatory – This method is must be implemented.

`Family.`**`use_isolated_family_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts retrievals to this family only.

compliance: mandatory – This method is must be implemented.

`Family.`**`can_search_relationships`**`()`

Tests if this user can perform `Relationship` searches.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer search operations to unauthorized users.

Returns `false` if search methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Family.`**`relationship_query`**

Gets a relationship query.

Returns the relationship query

Return type `osid.relationship.RelationshipQuery`

compliance: mandatory – This method must be implemented.

`Family.`**`get_relationships_by_query`**`(`*relationship_query*`)`

Gets a list of `Relationships` matching the given relationship query.

Parameters **`relationship_query`** (`osid.relationship.RelationshipQuery`) – the relationship query

Returns the returned `RelationshipList`

Return type `osid.relationship.RelationshipList`

Raise `NullArgument` – `relationship_query` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `relationship_query` is not of this service

compliance: mandatory – This method must be implemented.

Relationship Admin Methods

`Family.`**`family_id`**

Gets the `Familt Id` associated with this session.

Returns the `Family Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Family.`**`family`**

Gets the `Family` associated with this session.

Returns the family

Return type `osid.relationship.Family`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Family.`**`can_create_relationships`**`()`

Tests if this user can create `Relationships` A return of true does not guarantee successful authorization.

A return of false indicates that it is known creating a `Relationship` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer create operations to an unauthorized user.

Returns `false` if `Relationship` creation is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Family.`**`can_create_relationship_with_record_types`**`(`*relationship_record_types*`)`

Tests if this user can create a single `Relationship` using the desired record types.

While `RelationshipManager.getRelationshipRecordTypes()` can be used to examine which records are supported, this method tests which record(s) are required for creating a specific `Relationship`. Providing an empty array tests if a `Relationship` can be created with no records.

Parameters **`relationship_record_types`** (`osid.type.Type[]`) – array of relationship record types

Returns `true` if `Relationship` creation using the specified record `Types` is supported, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `relationship_record_types` is `null`

compliance: mandatory – This method must be implemented.

`Family.`**`get_relationship_form_for_create`**(*source_id*, *destination_id*, *relationship_record_types*)

Gets the relationship form for creating new relationships.

A new form should be requested for each create transaction.

Parameters

- **`source_id`** (`osid.id.Id`) – `Id` of a peer
- **`destination_id`** (`osid.id.Id`) – `Id` of the related peer
- **`relationship_record_types`** (`osid.type.Type[]`) – array of relationship record types

Returns the relationship form

Return type `osid.relationship.RelationshipForm`

Raise `NotFound` – `source_id` or `destination_id` is not found

Raise `NullArgument` – `source_id` or `destination_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – unable to get form for requested recod types

compliance: mandatory – This method must be implemented.

`Family.`**`create_relationship`**(*relationship_form*)

Creates a new `Relationship`.

Parameters **`relationship_form`** (`osid.relationship.RelationshipForm`) – the form for this `Relationship`

Returns the new `Relationship`

Return type `osid.relationship.Relationship`

Raise `IllegalState` – `relationship_form` already used in a create transaction

Raise `InvalidArgument` – one or more of the form elements is invalid

Raise `NullArgument` – `relationship_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `relationship_form` did not originate from `get_relationship_form_for_create()`

compliance: mandatory – This method must be implemented.

`Family.`**`can_update_relationships`**()

Tests if this user can update `Relationships`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating a `Relationship` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer update operations to an unauthorized user.

Returns `false` if `Relationship` modification is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Family.`**`get_relationship_form_for_update`**`(`*relationship_id*`)`

Gets the relationship form for updating an existing relationship.

A new relationship form should be requested for each update transaction.

Parameters **`relationship_id`** (`osid.id.Id`) – the `Id` of the `Relationship`

Returns the relationship form

Return type `osid.relationship.RelationshipForm`

Raise `NotFound` – `relationship_id` is not found

Raise `NullArgument` – `relationship_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Family.`**`update_relationship`**`(`*relationship_form*`)`

Updates an existing relationship.

Parameters **`relationship_form`** (`osid.relationship.RelationshipForm`) – the form containing the elements to be updated

Raise `IllegalState` – `relationship_form` already used in an update transaction

Raise `InvalidArgument` – the form contains an invalid value

Raise `NullArgument` – `relationship_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `relationship_form` did not originate from `get_relationship_form_for_update()`

compliance: mandatory – This method must be implemented.

`Family.`**`can_delete_relationships`**`()`

Tests if this user can delete `Relationships`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting a `Relationship` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer delete operations to an unauthorized user.

Returns `false` if `Relationship` deletion is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Family.`**`delete_relationship`**`(`*relationship_id*`)`

Deletes a `Relationship`.

Parameters **`relationship_id`** (`osid.id.Id`) – the `Id` of the `Relationship` to remove

Raise `NotFound` – `relationship_id` not found

Raise `NullArgument` – `relationship_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Family.**can_manage_relationship_aliases**()

Tests if this user can manage Id aliases for Relationships.

A return of true does not guarantee successful authorization. A return of false indicates that it is known changing an alias will result in a PermissionDenied. This is intended as a hint to an application that may opt not to offer alias operations to an unauthorized user.

Returns false if Relationship aliasing is not authorized, true otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

Family.**alias_relationship**(*relationship_id*, *alias_id*)

Adds an Id to a Relationship for the purpose of creating compatibility.

The primary Id of the Relationship is determined by the provider. The new Id performs as an alias to the primary Id. If the alias is a pointer to another relationship, it is reassigned to the given relationship Id.

Parameters

- **relationship_id** (osid.id.Id) – the Id of a Relationship
- **alias_id** (osid.id.Id) – the alias Id

Raise AlreadyExists – alias_id is already assigned

Raise NotFound – relationship not found

Raise NullArgument – relationship_id or alias_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Objects

Relationship

class dlkit.relationship.objects.**Relationship**

Bases: *dlkit.osid.objects.OsidRelationship*

A Relationship is an object between two peers.

The genus type indicates the relationship between the peer and the related peer.

source_id

Gets the from peer Id in this relationship.

Returns the peer

Return type osid.id.Id

compliance: mandatory – This method must be implemented.

destination_id

Gets the to peer Id in this relationship.

Returns the related peer

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`get_relationship_record`(*relationship_record_type*)

Gets the relationshop record corresponding to the given `Relationship` record `Type`.

This method is used to retrieve an object implementing the requested record. The `relationship_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(relationship_record_type)` is `true` .

Parameters **`relationship_record_type`** (`osid.type.Type`) – the type of relationship record to retrieve

Returns the relationship record

Return type `osid.relationship.records.RelationshipRecord`

Raise `NullArgument` – `relationship_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – `has_record_type(relationship_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Relationship Form

class `dlkit.relationship.objects.`**`RelationshipForm`**

Bases: *`dlkit.osid.objects.OsidRelationshipForm`*

This is the form for creating and updating `Relationships`.

Like all `OsidForm` objects, various data elements may be set here for use in the create and update methods in the `RelationshipAdminSession`. For each data element that may be set, metadata may be examined to provide display hints or data constraints.

`get_relationship_form_record`(*relationship_record_type*)

Gets the `RelationshipFormRecord` corresponding to the given relationship record `Type`.

Parameters **`relationship_record_type`** (`osid.type.Type`) – a relationship record type

Returns the relationship form record

Return type `osid.relationship.records.RelationshipFormRecord`

Raise `NullArgument` – `relationship_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – `has_record_type(relationship_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Relationship List

class `dlkit.relationship.objects.`**`RelationshipList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists, Relationship` provides a means for accessing `Relationship` elements sequentially either one at a time or many at a time.

Examples: while (rl.hasNext()) { Relationship relationship = rl.getNextRelationship(); }

or

while (rl.hasNext()) { Relationship[] relationships = rl.getNextRelationships(rl.available());

}

`next_relationship`

Gets the next `Relationship` in this list.

Returns the next `Relationship` in this list. The `has_next()` method should be used to test that a next `Relationship` is available before calling this method.

Return type `osid.relationship.Relationship`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_relationships`(*n*)

Gets the next set of `Relationships` elements in this list.

The specified amount must be less than or equal to the return from `available()`.

Parameters n (`cardinal`) – the number of `Relationship` elements requested which must be less than or equal to `available()`

Returns an array of `Relationship` elements.The length of the array is less than or equal to the number specified.

Return type `osid.relationship.Relationship`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Family

class `dlkit.relationship.objects.`**`Family`**(*abc_relationship_objects.Family*, *osid_objects.OsidCatalog*)

`:noindex:`

`get_family_record`(*family_record_type*)

Gets the famly record corresponding to the given `Family` record `Type`.

This method is used to retrieve an object implementing the requested record. The `family_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(family_record_type)` is `true`.

Parameters **`family_record_type`** (`osid.type.Type`) – the type of family record to retrieve

Returns the family record

Return type `osid.relationship.records.FamilyRecord`

Raise `NullArgument` – `family_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – `has_record_type(family_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Family Form

class `dlkit.relationship.objects.`**`FamilyForm`**

Bases: *`dlkit.osid.objects.OsidCatalogForm`*

This is the form for creating and updating `Family` objects.

Like all `OsidForm` objects, various data elements may be set here for use in the create and update methods in the `FamilyAdminSession`. For each data element that may be set, metadata may be examined to provide display hints or data constraints.

`get_family_form_record`(*family_record_type*)

Gets the `FamilyFormRecord` corresponding to the given family record `Type`.

Parameters **`family_record_type`** (`osid.type.Type`) – the family record type

Returns the family form record

Return type `osid.relationship.records.FamilyFormRecord`

Raise `NullArgument` – `family_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – `has_record_type(family_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Family List

class `dlkit.relationship.objects.`**`FamilyList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists, FamilyList` provides a means for accessing `Family` elements sequentially either one at a time or many at a time.

Examples: while (fl.hasNext()) { Family family = fl.getNextFamily(); }

or

while (fl.hasNext()) { Family[] families = fl.getNextFamilies(fl.available());

}

`next_family`

Gets the next `Family` in this list.

Returns the next `Family` in this list. The `has_next()` method should be used to test that a next `Family` is available before calling this method.

Return type `osid.relationship.Family`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_families`(*n*)

Gets the next set of `Family elements` in this list.

The specified amount must be less than or equal to the return from `available()`.

Parameters **`n`** (`cardinal`) – the number of `Family` elements requested which must be less than or equal to `available()`

Returns an array of `Family` elements.The length of the array is less than or equal to the number specified.

Return type `osid.relationship.Family`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Family Node

class `dlkit.relationship.objects.`**`FamilyNode`**

Bases: *`dlkit.osid.objects.OsidNode`*

This interface is a container for a partial hierarchy retrieval.

The number of hierarchy levels traversable through this interface depend on the number of levels requested in the `FamilyHierarchySession`.

`family`

Gets the `Family` at this node.

Returns the family represented by this node

Return type `osid.relationship.Family`

compliance: mandatory – This method must be implemented.

`parent_family_nodes`

Gets the parents of this family.

Returns the parents of the `id`

Return type `osid.relationship.FamilyNodeList`

compliance: mandatory – This method must be implemented.

`child_family_nodes`

Gets the children of this family.

Returns the children of this family

Return type `osid.relationship.FamilyNodeList`

compliance: mandatory – This method must be implemented.

Family Node List

class `dlkit.relationship.objects.`**`FamilyNodeList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists, FamilyNodeList` provides a means for accessing `FamilyNode` elements sequentially either one at a time or many at a time.

Examples: while (fnl.hasNext()) { FamilyNode node = fnl.getNextFamilyNode(); }

or

while (fnl.hasNext()) { FamilyNode[] nodes = fnl.getNextFamilyNodes(fnl.available());

}

`next_family_node`

Gets the next `FamilyNode` in this list.

Returns the next `FamilyNode` in this list. The `has_next()` method should be used to test that a next `FamilyNode` is available before calling this method.

Return type `osid.relationship.FamilyNode`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_family_nodes`(*n*)

Gets the next set of `FamilyNode elements` in this list.

The specified amount must be less than or equal to the return from `available()`.

Parameters **n** (`cardinal`) – the number of `FamilyNode` elements requested which must be less than or equal to `available()`

Returns an array of `FamilyNode` elements.The length of the array is less than or equal to the number specified.

Return type `osid.relationship.FamilyNode`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Queries

Relationship Query

class `dlkit.relationship.queries.`**`RelationshipQuery`**

Bases: *`dlkit.osid.queries.OsidRelationshipQuery`*

This is the query for searching relationships.

Each method match specifies an `AND` term while multiple invocations of the same method produce a nested `OR`.

`match_source_id`(*peer*, *match*)

Matches a relationship peer.

Parameters

- **peer** (osid.id.Id) – peer Id to match
- **match** (boolean) – true for a positive match, false for a negative match

Raise NullArgument – peer is null

compliance: mandatory – This method must be implemented.

source_id_terms

match_destination_id(*peer*, *match*)

Matches the other relationship peer.

Parameters

- **peer** (osid.id.Id) – peer Id to match
- **match** (boolean) – true for a positive match, false for a negative match

Raise NullArgument – peer is null

compliance: mandatory – This method must be implemented.

destination_id_terms

match_same_peer_id(*match*)

Matches circular relationships to the same peer.

Parameters **match** (boolean) – true for a positive match, false for a negative match

compliance: mandatory – This method must be implemented.

same_peer_id_terms

match_family_id(*family_id*, *match*)

Sets the family Id for this query.

Parameters

- **family_id** (osid.id.Id) – a family Id
- **match** (boolean) – true for a positive match, false for a negative match

Raise NullArgument – family_id is null

compliance: mandatory – This method must be implemented.

family_id_terms

supports_family_query()

Tests if a FamilyQuery is available.

Returns true if a family query is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

family_query

Gets the query for a family.

Multiple retrievals produce a nested OR term.

Returns the family query

Return type osid.relationship.FamilyQuery

Raise Unimplemented – supports_family_query() is false

compliance: optional – This method must be implemented if ``supports_family_query()`` is ``true``.

family_terms

get_relationship_query_record(*relationship_record_type*)
Gets the relationship query record corresponding to the given `Relationship` record `Type`.

Multiple record retrievals produce a nested `OR` term.

Parameters **relationship_record_type** (`osid.type.Type`) – a relationship record type

Returns the relationship query record

Return type `osid.relationship.records.RelationshipQueryRecord`

Raise `NullArgument` – `relationship_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – `has_record_type(relationship_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Family Query

class `dlkit.relationship.queries.`**FamilyQuery**
Bases: *`dlkit.osid.queries.OsidCatalogQuery`*

This is the query interface for searching for families.

Each method specifies an `AND` term while multiple invocations of the same method produce a nested `OR`.

match_relationship_id(*relationship_id, match*)
Matches a relationship `Id`.

Parameters

- **relationship_id** (`osid.id.Id`) – a relationship `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `relationship_id` is `null`

compliance: mandatory – This method must be implemented.

relationship_id_terms

supports_relationship_query()
Tests if a relationship query is available.

Returns `true` if a relationship query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

relationship_query
Gets the query interface for a relationship.

Returns the relationship query

Return type `osid.relationship.RelationshipQuery`

Raise `Unimplemented` – `supports_relationship_query()` is `false` *compliance: optional – This method must be implemented if ‘‘supports_relationship_query()‘‘ is ‘‘true‘‘.*

match_any_relationship(*match*)

Matches families with any relationship.

Parameters match (boolean) – true to match families with any relationship, false to match families with no relationship

compliance: mandatory – This method must be implemented.

relationship_terms

match_ancestor_family_id(*family_id*, *match*)

Sets the family Id for this query to match families that have the specified family as an ancestor.

Parameters

- **family_id** (osid.id.Id) – a family Id
- **match** (boolean) – true for a positive match, false for a negative match

Raise NullArgument – family_id is null

compliance: mandatory – This method must be implemented.

ancestor_family_id_terms

supports_ancestor_family_query()

Tests if a FamilyQuery is available.

Returns true if a family query interface is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

ancestor_family_query

Gets the query interface for a family.

Multiple retrievals produce a nested OR term.

Returns the family query

Return type osid.relationship.FamilyQuery

Raise Unimplemented – supports_ancestor_family_query() is false

compliance: optional – This method must be implemented if ‘‘supports_ancestor_family_query()‘‘ is ‘‘true‘‘.

match_any_ancestor_family(*match*)

Matches families with any ancestor.

Parameters match (boolean) – true to match families with any ancestor, false to match root families

compliance: mandatory – This method must be implemented.

ancestor_family_terms

match_descendant_family_id(*family_id*, *match*)

Sets the family Id for this query to match families that have the specified family as a descednant.

Parameters

- **family_id** (osid.id.Id) – a family Id
- **match** (boolean) – true for a positive match, false for a negative match

Raise `NullArgument` – `family_id` is `null`

compliance: mandatory – This method must be implemented.

`descendant_family_id_terms`

`supports_descendant_family_query``()`
Tests if a `FamilyQuery` is available.

Returns `true` if a family query interface is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`descendant_family_query`
Gets the query interface for a family.

Multiple retrievals produce a nested `OR` term.

Returns the family query

Return type `osid.relationship.FamilyQuery`

Raise `Unimplemented` – `supports_descendant_family_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_descendant_family_query()‘‘ is ‘‘true‘‘.

`match_any_descendant_family``(`*match*`)`
Matches families with any decendant.

Parameters **`match`** (`boolean`) – `true` to match families with any decendants, `false` to match leaf families

compliance: mandatory – This method must be implemented.

`descendant_family_terms`

`get_family_query_record``(`*family_record_type*`)`
Gets the family query record corresponding to the given `Family` record `Type`.

Multiple record retrievals produce a nested boolean `OR` term.

Parameters **`family_record_type`** (`osid.type.Type`) – a family record type

Returns the family query record

Return type `osid.relationship.records.FamilyQueryRecord`

Raise `NullArgument` – `family_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure occurred

Raise `Unsupported` – `has_record_type(family_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Records

Relationship Record

class `dlkit.relationship.records.`**`RelationshipRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `Relationship`.

The methods specified by the record type are available through the underlying object.

Relationship Query Record

class `dlkit.relationship.records.`**`RelationshipQueryRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `RelationshipQuery`.

The methods specified by the record type are available through the underlying object.

Relationship Form Record

class `dlkit.relationship.records.`**`RelationshipFormRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `RelationshipForm`.

The methods specified by the record type are available through the underlying object.

Relationship Search Record

class `dlkit.relationship.records.`**`RelationshipSearchRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `RelationshipSearch`.

The methods specified by the record type are available through the underlying object.

Family Record

class `dlkit.relationship.records.`**`FamilyRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `Family`.

The methods specified by the record type are available through the underlying object.

Family Query Record

class `dlkit.relationship.records.`**`FamilyQueryRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `FamilyQuery`.

The methods specified by the record type are available through the underlying object.

Family Form Record

class `dlkit.relationship.records.`**`FamilyFormRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `FamilyForm`.

The methods specified by the record type are available through the underlying object.

Family Search Record

class dlkit.relationship.records.**FamilySearchRecord**

Bases: *dlkit.osid.records.OsidRecord*

A record for a FamilySearch.

The methods specified by the record type are available through the underlying object.

Repository

Summary

Repository Open Service Interface Definitions repository version 3.0.0

The Repository OSID provides the service of finding and managing digital assets.

Assets

An Asset represents a unit of content, whether it be an image, a video, an application document or some text. The Asset defines a core set of definitions applicable to digital content, such as copyright and publisher, and allows for a type specification to be appended as with other OsidObjects.

Asset content, such as a document, is defined such that there may be multiple formats contained with the same asset. A document may be accessible in both PDF and MS Word, but is the same document, for example. An image may have both a large size and a thumbnail version. Generally, an asset contains more than one version of content when it is left to the application to decide which is most appropriate.

The Asset Type may define methods in common throughout the content variations. An example asset is one whose content Types are "Quicktime" and "MPEG", but the Asset Type is "movie" and defines methods that describe the move aside from the formats. This "double" Type hierarchy stemming from the asset requires more care in defining interfaces.

Assets also have "credits" which define the authors, editors, creators, performers, producers or any other "role", identified with a role Type, with the production of the asset. These are managed externally to the asset through another OsidSession.

Through additional optional OsidSessions, the Asset can be "extended" to offer temporal information. An asset may pertain to a date, a period of time, or a series of dates and periods. This mechanism is to offer the ability to search for assets pertaining to a desired date range without requiring understanding of a Type.

Similarly, the Asset can also map to spatial information. A photograph may be "geotagged" with the GPS coordinates where it was taken, a conical shape in stellar coordinates could be described for an astronimocal image, or there may be a desire to may a historical book to the spatial coordinates of Boston and Philadelphia. Unlike temporal mappings, the definition of the spatial coordinate is left to a spatial Type to define. The Repository OSID simply manages spatial mappings to the Asset.

Asset Tagging

Assets may also relate to Ontology OSID Subjects. The Subject provides the ability to normalize information related to subject matter across the Assets to simplify management and provide a more robust searching mechanism. For example, with a photograph of the Empire State Building, one may wish to describe that it was designed by Shreve, Lamb and Harmon and completed in 1931. The information about the building itself can be described using a Subject and related to the photograph, and any other photograph that captures the building. The Asset Type for the photograph may simply be "photograph" and doesn't attempt to describe a building, while the `AssetContent Type` is "image/jpeg".

An application performing a search for Empire State Building can be execute the search over the `Subjects,` and once the user has narrowed the subject area, then the related Assets can be retrieved, and from there negotiate the content.

A provider wishing to construct a simple inventory database of buildings in New York may decide to do so using the Resource OSID. The `Resource Type` may describe the construction dates, height, location, style and architects of buildings. The `Type` may also include a means of getting a reference image using the `Asset` interface. Since there is no explicit relationship between `Subject` and `Resource,` the `Resource` can be adapted to the `Subject` interface (mapping a `building_resource_type` to a `building_subject_type`) to use the same data for `Subject` to `Asset` mappings and searching.

Asset Compositions

Asset compositions can be created using the `Composition` interface. A `Composition` is a group of `Assets` and compositions may be structured into a hierarchy for the purpose of "building" larger content. A content management system may make use of this interface to construct a web page. The `Composition` hierarchy may map into an XHTML structure and each `Asset` represent an image or a link in the document. However, the produced web page at a given URL may be represented by another single `Asset` that whose content has both the URL and the XHTML stream.

Another example is an IMS Common Cartridge. The `Composition` may be used to produce the zip file cartridge, but consumers may access the zip file via an `Asset` .

Repository Cataloging

Finally, `Assets` and `Compositions` may be categorized into `Repository` objects. A `Repository` is a catalog-like interface to help organize assets and subject matter. Repositories may be organized into hierarchies for organization or federation purposes.

This number of service aspects to this Repository OSID produce a large number of definitions. It is recommended to use the `RepositoryManager` definition to select a single `OsidSession` of interest, and work that definition through its dependencies before tackling another aspect.

Sub Packages

The Repository OSID includes a rules subpackage for managing dynamic compositions. Repository Open Service Interface Definitions repository version 3.0.0

The Repository OSID provides the service of finding and managing digital assets.

Assets

An `Asset` represents a unit of content, whether it be an image, a video, an application document or some text. The Asset defines a core set of definitions applicable to digital content, such as copyright and publisher, and allows for a type specification to be appended as with other `OsidObjects`.

Asset content, such as a document, is defined such that there may be multiple formats contained with the same asset. A document may be accessible in both PDF and MS Word, but is the same document, for example. An image may have both a large size and a thumbnail version. Generally, an asset contains more than one version of content when it is left to the application to decide which is most appropriate.

The `Asset Type` may define methods in common throughout the content variations. An example asset is one whose content `Types` are "Quicktime" and "MPEG", but the `Asset Type` is "movie" and defines methods that describe the move aside from the formats. This "double" Type hierarchy stemming from the asset requires more care in defining interfaces.

`Assets` also have "credits" which define the authors, editors, creators, performers, producers or any other "role", identified with a role `Type,` with the production of the asset. These are managed externally to the asset through another `OsidSession`.

Through additional optional `OsidSessions,` the `Asset` can be "extended" to offer temporal information. An asset may pertain to a date, a period of time, or a series of dates and periods. This mechanism is to offer the ability to search for assets pertaining to a desired date range without requiring understanding of a `Type`.

Similarly, the `Asset` can also map to spatial information. A photograph may be "geotagged" with the GPS coordinates where it was taken, a conical shape in stellar coordinates could be described for an astronimocal image, or there may be a desire to may a historical book to the spatial coordinates of Boston and Philadelphia. Unlike temporal mappings, the definition of the spatial coordinate is left to a spatial Type to define. The Repository OSID simply manages spatial mappings to the Asset.

Asset Tagging

`Assets` may also relate to Ontology OSID `Subjects`. The `Subject` provides the ability to normalize information related to subject matter across the `Assets` to simplify management and provide a more robust searching mechanism. For example, with a photograph of the Empire State Building, one may wish to describe that it was designed by Shreve, Lamb and Harmon and completed in 1931. The information about the building itself can be described using a `Subject` and related to the photograph, and any other photograph that captures the building. The `Asset Type` for the photograph may simply be "photograph" and doesn't attempt to describe a building, while the `AssetContent Type` is "image/jpeg".

An application performing a search for Empire State Building can be execute the search over the `Subjects,` and once the user has narrowed the subject area, then the related Assets can be retrieved, and from there negotiate the content.

A provider wishing to construct a simple inventory database of buildings in New York may decide to do so using the Resource OSID. The `Resource Type` may describe the construction dates, height, location, style and architects of buildings. The `Type` may also include a means of getting a reference image using the `Asset` interface. Since there is no explicit relationship between `Subject` and `Resource,` the `Resource` can be adapted to the `Subject` interface (mapping a `building_resource_type` to a `building_subject_type`) to use the same data for `Subject` to `Asset` mappings and searching.

Asset Compositions

Asset compositions can be created using the `Composition` interface. A `Composition` is a group of `Assets` and compositions may be structured into a hierarchy for the purpose of "building" larger content. A content management system may make use of this interface to construct a web page. The `Composition` hierarchy may map into an XHTML structure and each `Asset` represent an image or a link in the document. However, the produced web page at a given URL may be represented by another single `Asset` that whose content has both the URL and the XHTML stream.

Another example is an IMS Common Cartridge. The `Composition` may be used to produce the zip file cartridge, but consumers may access the zip file via an `Asset` .

Repository Cataloging

Finally, `Assets` and `Compositions` may be categorized into `Repository` objects. A `Repository` is a catalog-like interface to help organize assets and subject matter. Repositories may be organized into hierarchies for organization or federation purposes.

This number of service aspects to this Repository OSID produce a large number of definitions. It is recommended to use the `RepositoryManager` definition to select a single `OsidSession` of interest, and work that definition through its dependencies before tackling another aspect.

Sub Packages

The Repository OSID includes a rules subpackage for managing dynamic compositions.

Service Managers

Repository Profile

class `dlkit.services.repository.`**`RepositoryProfile`**

Bases: `dlkit.osid.managers.OsidProfile`

The repository profile describes interoperability among repository services.

`supports_asset_lookup``()`

Tests if asset lookup is supported.

Returns `true` if asset lookup is supported , `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_asset_query``()`

Tests if asset query is supported.

Returns `true` if asset query is supported , `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_asset_search``()`

Tests if asset search is supported.

Returns `true` if asset search is supported , `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_asset_admin``()`

Tests if asset administration is supported.

Returns `true` if asset administration is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_asset_notification``()`

Tests if asset notification is supported.

A repository may send messages when assets are created, modified, or deleted.

Returns `true` if asset notification is supported , `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_asset_repository``()`

Tests if retrieving mappings of assets and repositories is supported.

Returns `true` if asset repository mapping retrieval is supported , `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_asset_repository_assignment``()`

Tests if managing mappings of assets and repositories is supported.

Returns `true` if asset repository assignment is supported , `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_asset_composition()
Tests if assets are included in compositions.

Returns `true` if asset composition supported , `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_asset_composition_design()
Tests if mapping assets to compositions is supported.

Returns `true` if designing asset compositions is supported , `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_composition_lookup()
Tests if composition lookup is supported.

Returns `true` if composition lookup is supported , `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_composition_query()
Tests if composition query is supported.

Returns `true` if composition query is supported , `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_composition_search()
Tests if composition search is supported.

Returns `true` if composition search is supported , `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_composition_admin()
Tests if composition administration is supported.

Returns `true` if composition administration is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_composition_repository()
Tests if retrieval of composition to repository mappings is supported.

Returns `true` if composition to repository mapping is supported , `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_composition_repository_assignment()
Tests if assigning composition to repository mappings is supported.

Returns `true` if composition to repository assignment is supported , `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_repository_lookup`()
Tests if repository lookup is supported.

Returns `true` if repository lookup is supported , `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_repository_query`()
Tests if repository query is supported.

Returns `true` if repository query is supported , `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_repository_admin`()
Tests if repository administration is supported.

Returns `true` if repository administration is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_repository_hierarchy`()
Tests if a repository hierarchy traversal is supported.

Returns `true` if a repository hierarchy traversal is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_repository_hierarchy_design`()
Tests if a repository hierarchy design is supported.

Returns `true` if a repository hierarchy design is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`asset_record_types`
Gets all the asset record types supported.

Returns the list of supported asset record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

`asset_search_record_types`
Gets all the asset search record types supported.

Returns the list of supported asset search record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

asset_content_record_types

Gets all the asset content record types supported.

Returns the list of supported asset content record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

composition_record_types

Gets all the composition record types supported.

Returns the list of supported composition record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

composition_search_record_types

Gets all the composition search record types supported.

Returns the list of supported composition search record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

repository_record_types

Gets all the repository record types supported.

Returns the list of supported repository record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

repository_search_record_types

Gets all the repository search record types supported.

Returns the list of supported repository search record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

spatial_unit_record_types

Gets all the spatial unit record types supported.

Returns the list of supported spatial unit record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

coordinate_types

Gets all the coordinate types supported.

Returns the list of supported coordinate types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

Repository Manager

class `dlkit.services.repository.`**`RepositoryManager`**`(`*proxy=None*`)`

Bases: `dlkit.osid.managers.OsidManager`, `dlkit.osid.sessions.OsidSession`, *`dlkit.services.repository.RepositoryProfile`*

The repository manager provides access to asset lookup and creation session and provides interoperability tests for various aspects of this service.

The sessions included in this manager are:

- `AssetLookupSession`: a session to retrieve assets
- `AssetQuerySession`: a session to query assets
- `AssetSearchSession`: a session to search for assets
- `AssetAdminSession`: a session to create and delete assets
- `AssetNotificationSession`: a session to receive notifications pertaining to asset changes
- `AssetRepositorySession`: a session to look up asset to repository mappings
- `AssetRepositoryAssignmentSession`: a session to manage asset to repository mappings
- `AssetSmartRepositorySession`: a session to manage dynamic repositories of assets
- `AssetTemporalSession`: a session to access the temporal coverage of an asset
- `AssetTemporalAssignmentSession`: a session to manage the temporal coverage of an asset
- `AssetSpatialSession`: a session to access the spatial coverage of an asset
- `AssetSpatialAssignmentSession`: a session to manage the spatial coverage of an asset
- `AssetCompositionSession`: a session to look up asset composition mappings
- `AssetCompositionDesignSession`: a session to map assets to compositions
- `CompositionLookupSession`: a session to retrieve compositions
- `CompositionQuerySession`: a session to query compositions
- `CompositionSearchSession`: a session to search for compositions
- `CompositionAdminSession`: a session to create, update and delete compositions
- `CompositionNotificationSession`: a session to receive notifications pertaining to changes in compositions
- `CompositionRepositorySession`: a session to retrieve composition repository mappings
- `CompositionRepositoryAssignmentSession`: a session to manage composition repository mappings
- `CompositionSmartRepositorySession`: a session to manage dynamic repositories of compositions
- `RepositoryLookupSession`: a session to retrieve repositories
- `RepositoryQuerySession`: a session to query repositories
- `RepositorySearchSession`: a session to search for repositories
- `RepositoryAdminSession`: a session to create, update and delete repositories
- `RepositoryNotificationSession`: a session to receive notifications pertaining to changes in repositories

- `RepositoryHierarchySession`: a session to traverse repository hierarchies
- `RepositoryHierarchyDesignSession`: a session to manage repository hierarchies

`repository_batch_manager`
Gets a `RepositoryBatchManager`.

Returns a `RepostoryBatchManager`

Return type `osid.repository.batch.RepositoryBatchManager`

Raise `OperationFailed` – unable to complete request

Raise `Unimplemented` – `supports_repository_batch()` is `false`

compliance: optional – This method must be implemented if ``supports_repository_batch()`` is ``true``.

`repository_rules_manager`
Gets a `RepositoryRulesManager`.

Returns a `RepostoryRulesManager`

Return type `osid.repository.rules.RepositoryRulesManager`

Raise `OperationFailed` – unable to complete request

Raise `Unimplemented` – `supports_repository_rules()` is `false`

compliance: optional – This method must be implemented if ``supports_repository_rules()`` is ``true``.

Repository Lookup Methods

`RepositoryManager.`**`can_lookup_repositories`**`()`
Tests if this user can perform `Repository` lookups.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if lookup methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`use_comparative_repository_view`**`()`
The returns from the repository methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`RepositoryManager.`**`use_plenary_repository_view`**`()`
A complete view of the `Repository` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`RepositoryManager.`**`get_repository`**`(`*repository_id*`)`
Gets the `Repository` specified by its `Id`.

In plenary mode, the exact `Id` is found or a `NotFound` results. Otherwise, the returned `Repository` may have a different `Id` than requested, such as the case where a duplicate `Id` was assigned to a `Repository` and retained for compatibility.

Parameters **`repository_id`** (`osid.id.Id`) – `Id` of the `Repository`

Returns the repository

Return type `osid.repository.Repository`

Raise `NotFound` – `repository_id` not found

Raise `NullArgument` – `repository_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method is must be implemented.

`RepositoryManager.`**`get_repositories_by_ids`**(*repository_ids*)

Gets a `RepositoryList` corresponding to the given `IdList`.

In plenary mode, the returned list contains all of the repositories specified in the `Id` list, in the order of the list, including duplicates, or an error results if an `Id` in the supplied list is not found or inaccessible. Otherwise, inaccessible `Repositories` may be omitted from the list and may present the elements in any order including returning a unique set.

Parameters **`repository_ids`** (`osid.id.IdList`) – the list of `Ids` to retrieve

Returns the returned `Repository list`

Return type `osid.repository.RepositoryList`

Raise `NotFound` – an `Id` was not found

Raise `NullArgument` – `repository_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`get_repositories_by_genus_type`**(*repository_genus_type*)

Gets a `RepositoryList` corresponding to the given repository genus `Type` which does not include repositories of types derived from the specified `Type`.

In plenary mode, the returned list contains all known repositories or an error results. Otherwise, the returned list may contain only those repositories that are accessible through this session.

Parameters **`repository_genus_type`** (`osid.type.Type`) – a repository genus type

Returns the returned `Repository list`

Return type `osid.repository.RepositoryList`

Raise `NullArgument` – `repository_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

RepositoryManager.**get_repositories_by_parent_genus_type**(*repository_genus_type*)
Gets a `RepositoryList` corresponding to the given repository genus `Type` and include any additional repositories with genus types derived from the specified `Type`.

In plenary mode, the returned list contains all known repositories or an error results. Otherwise, the returned list may contain only those repositories that are accessible through this session.

Parameters **repository_genus_type** (`osid.type.Type`) – a repository genus type

Returns the returned `Repository list`

Return type `osid.repository.RepositoryList`

Raise `NullArgument` – `repository_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

RepositoryManager.**get_repositories_by_record_type**(*repository_record_type*)
Gets a `RepositoryList` containing the given repository record `Type`.

In plenary mode, the returned list contains all known repositories or an error results. Otherwise, the returned list may contain only those repositories that are accessible through this session.

Parameters **repository_record_type** (`osid.type.Type`) – a repository record type

Returns the returned `Repository list`

Return type `osid.repository.RepositoryList`

Raise `NullArgument` – `repository_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

RepositoryManager.**get_repositories_by_provider**(*resource_id*)
Gets a `RepositoryList` from the given provider ““.

In plenary mode, the returned list contains all known repositories or an error results. Otherwise, the returned list may contain only those repositories that are accessible through this session.

Parameters **resource_id** (`osid.id.Id`) – a resource `Id`

Returns the returned `Repository list`

Return type `osid.repository.RepositoryList`

Raise `NullArgument` – `resource_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

RepositoryManager.**repositories**
Gets all `Repositories`.

In plenary mode, the returned list contains all known repositories or an error results. Otherwise, the returned list may contain only those repositories that are accessible through this session.

Returns a list of `Repositories`

Return type `osid.repository.RepositoryList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Repository Query Methods

`RepositoryManager.`**`can_search_repositories`**`()`

Tests if this user can perform `Repository` searches.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer search operations to unauthorized users.

Returns `false` if search methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`repository_query`**

Gets a repository query.

Returns the repository query

Return type `osid.repository.RepositoryQuery`

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`get_repositories_by_query`**`(`*repository_query*`)`

Gets a list of `Repositories` matching the given repository query.

Parameters **`repository_query`** (`osid.repository.RepositoryQuery`) – the repository query

Returns the returned `RepositoryList`

Return type `osid.repository.RepositoryList`

Raise `NullArgument` – `repository_query` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `repository_query` is not of this service

compliance: mandatory – This method must be implemented.

Repository Admin Methods

`RepositoryManager.`**`can_create_repositories`**`()`

Tests if this user can create `Repositories`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known creating a `Repository` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer create operations to unauthorized users.

Returns `false` if `Repository` creation is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`can_create_repository_with_record_types`**(*repository_record_types*)

Tests if this user can create a single `Repository` using the desired record types.

While `RepositoryManager.getRepositoryRecordTypes()` can be used to examine which records are supported, this method tests which record(s) are required for creating a specific `Repository`. Providing an empty array tests if a `Repository` can be created with no records.

Parameters **`repository_record_types`** (`osid.type.Type[]`) – array of repository record types

Returns `true` if `Repository` creation using the specified `Types` is supported, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `repository_record_types` is `null`

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`get_repository_form_for_create`**(*repository_record_types*)

Gets the repository form for creating new repositories.

A new form should be requested for each create transaction.

Parameters **`repository_record_types`** (`osid.type.Type[]`) – array of repository record types

Returns the repository form

Return type `osid.repository.RepositoryForm`

Raise `NullArgument` – `repository_record_types` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – unable to get form for requested record types

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`create_repository`**(*repository_form*)

Creates a new `Repository`.

Parameters **`repository_form`** (`osid.repository.RepositoryForm`) – the form for this `Repository`

Returns the new `Repository`

Return type `osid.repository.Repository`

Raise `IllegalState` – `repository_form` already used in a create transaction

Raise `InvalidArgument` – one or more of the form elements is invalid

Raise `NullArgument` – `repository_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `repository_form` did not originate from `get_repository_form_for_create()`

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`can_update_repositories`**`()`

Tests if this user can update `Repositories`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating a `Repository` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer update operations to unauthorized users.

Returns `false` if `Repository` modification is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`get_repository_form_for_update`**`(`*repository_id*`)`

Gets the repository form for updating an existing repository.

A new repository form should be requested for each update transaction.

Parameters **`repository_id`** (`osid.id.Id`) – the `Id` of the `Repository`

Returns the repository form

Return type `osid.repository.RepositoryForm`

Raise `NotFound` – `repository_id` is not found

Raise `NullArgument` – `repository_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`update_repository`**`(`*repository_form*`)`

Updates an existing repository.

Parameters **`repository_form`** (`osid.repository.RepositoryForm`) – the form containing the elements to be updated

Raise `IllegalState` – `repository_form` already used in an update transaction

Raise `InvalidArgument` – the form contains an invalid value

Raise `NullArgument` – `repository_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `repository_form` did not originate from `get_repository_form_for_update()`

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`can_delete_repositories`**`()`

Tests if this user can delete `Repositories`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting a `Repository` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer delete operations to unauthorized users.

Returns `false` if `Repository` deletion is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

RepositoryManager.**delete_repository**(*repository_id*)

Deletes a Repository.

Parameters **repository_id** (osid.id.Id) – the Id of the Repository to remove

Raise NotFound – repository_id not found

Raise NullArgument – repository_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

RepositoryManager.**can_manage_repository_aliases**()

Tests if this user can manage Id aliases for repositories.

A return of true does not guarantee successful authorization. A return of false indicates that it is known changing an alias will result in a PermissionDenied. This is intended as a hint to an application that may opt not to offer alias operations to an unauthorized user.

Returns false if Repository aliasing is not authorized, true otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

RepositoryManager.**alias_repository**(*repository_id*, *alias_id*)

Adds an Id to a Repository for the purpose of creating compatibility.

The primary Id of the Repository is determined by the provider. The new Id is an alias to the primary Id. If the alias is a pointer to another repository, it is reassigned to the given repository Id.

Parameters

- **repository_id** (osid.id.Id) – the Id of a Repository
- **alias_id** (osid.id.Id) – the alias Id

Raise AlreadyExists – alias_id is in use as a primary Id

Raise NotFound – repository_id not found

Raise NullArgument – repository_id or alias_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Repository Hierarchy Methods

RepositoryManager.**repository_hierarchy_id**

Gets the hierarchy Id associated with this session.

Returns the hierarchy Id associated with this session

Return type osid.id.Id

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`repository_hierarchy`**

Gets the hierarchy associated with this session.

Returns the hierarchy associated with this session

Return type `osid.hierarchy.Hierarchy`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`can_access_repository_hierarchy`**`()`

Tests if this user can perform hierarchy queries.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations.

Returns `false` if hierarchy traversal methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`use_comparative_repository_view`**`()`

The returns from the repository methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`RepositoryManager.`**`use_plenary_repository_view`**`()`

A complete view of the `Repository` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`RepositoryManager.`**`root_repository_ids`**

Gets the root repository `Ids` in this hierarchy.

Returns the root repository `Ids`

Return type `osid.id.IdList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`root_repositories`**

Gets the root repositories in the repository hierarchy.

A node with no parents is an orphan. While all repository `Ids` are known to the hierarchy, an orphan does not appear in the hierarchy unless explicitly added as a root node or child of another node.

Returns the root repositories

Return type `osid.repository.RepositoryList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method is must be implemented.

`RepositoryManager.`**`has_parent_repositories`**`(`*repository_id*`)`

Tests if the `Repository` has any parents.

Parameters **`repository_id`** (`osid.id.Id`) – a repository `Id`

Returns `true` if the repository has parents, `false` otherwise

Return type `boolean`

Raise `NotFound` – `repository_id` is not found

Raise `NullArgument` – `repository_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`is_parent_of_repository`**`(`*id_*, *repository_id*`)`

Tests if an `Id` is a direct parent of a repository.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`repository_id`** (`osid.id.Id`) – the `Id` of a repository

Returns `true` if this `id` is a parent of `repository_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `repository_id` is not found

Raise `NullArgument` – `id` or `repository_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` not found return `false`.

`RepositoryManager.`**`get_parent_repository_ids`**`(`*repository_id*`)`

Gets the parent `Ids` of the given repository.

Parameters **`repository_id`** (`osid.id.Id`) – a repository `Id`

Returns the parent `Ids` of the repository

Return type `osid.id.IdList`

Raise `NotFound` – `repository_id` is not found

Raise `NullArgument` – `repository_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`get_parent_repositories`**`(`*repository_id*`)`

Gets the parents of the given repository.

Parameters **`repository_id`** (`osid.id.Id`) – the `Id` to query

Returns the parents of the repository

Return type `osid.repository.RepositoryList`

Raise `NotFound` – `repository_id` not found

Raise `NullArgument` – `repository_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`is_ancestor_of_repository`**(*id_*, *repository_id*)

Tests if an `Id` is an ancestor of a repository.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`repository_id`** (`osid.id.Id`) – the Id of a repository

Returns `true` if this `id` is an ancestor of `repository_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `repository_id` not found

Raise `NullArgument` – `repository_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` not found return `false`.

`RepositoryManager.`**`has_child_repositories`**(*repository_id*)

Tests if a repository has any children.

Parameters **`repository_id`** (`osid.id.Id`) – a repository `Id`

Returns `true` if the `repository_id` has children, `false` otherwise

Return type `boolean`

Raise `NotFound` – `repository_id` not found

Raise `NullArgument` – `repository_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`is_child_of_repository`**(*id_*, *repository_id*)

Tests if a node is a direct child of another.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`repository_id`** (`osid.id.Id`) – the `Id` of a repository

Returns `true` if the `id` is a child of `repository_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `repository_id` not found

Raise `NullArgument` – `repository_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` not found return `false`.

`RepositoryManager.`**`get_child_repository_ids`**`(`*repository_id*`)`

Gets the `Ids` of the children of the given repository.

Parameters **`repository_id`** (`osid.id.Id`) – the `Id` to query

Returns the children of the repository

Return type `osid.id.IdList`

Raise `NotFound` – `repository_id` not found

Raise `NullArgument` – `repository_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`get_child_repositories`**`(`*repository_id*`)`

Gets the children of the given repository.

Parameters **`repository_id`** (`osid.id.Id`) – the `Id` to query

Returns the children of the repository

Return type `osid.repository.RepositoryList`

Raise `NotFound` – `repository_id` not found

Raise `NullArgument` – `repository_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`is_descendant_of_repository`**`(`*id_*, *repository_id*`)`

Tests if an `Id` is a descendant of a repository.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`repository_id`** (`osid.id.Id`) – the `Id` of a repository

Returns `true` if the `id` is a descendant of the `repository_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `repository_id` not found

Raise `NullArgument` – `repository_id` or `id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` is not found return `false`.

RepositoryManager.**get_repository_node_ids**(*repository_id*, *ancestor_levels*, *descendant_levels*, *include_siblings*)

Gets a portion of the hierarchy for the given repository.

Parameters

- **repository_id** (osid.id.Id) – the Id to query
- **ancestor_levels** (cardinal) – the maximum number of ancestor levels to include. A value of 0 returns no parents in the node.
- **descendant_levels** (cardinal) – the maximum number of descendant levels to include. A value of 0 returns no children in the node.
- **include_siblings** (boolean) – true to include the siblings of the given node, false to omit the siblings

Returns the specified repository node

Return type osid.hierarchy.Node

Raise NotFound – repository_id not found

Raise NullArgument – repository_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

RepositoryManager.**get_repository_nodes**(*repository_id*, *ancestor_levels*, *descendant_levels*, *include_siblings*)

Gets a portion of the hierarchy for the given repository.

Parameters

- **repository_id** (osid.id.Id) – the Id to query
- **ancestor_levels** (cardinal) – the maximum number of ancestor levels to include. A value of 0 returns no parents in the node.
- **descendant_levels** (cardinal) – the maximum number of descendant levels to include. A value of 0 returns no children in the node.
- **include_siblings** (boolean) – true to include the siblings of the given node, false to omit the siblings

Returns the specified repository node

Return type osid.repository.RepositoryNode

Raise NotFound – repository_id not found

Raise NullArgument – repository_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Repository Hierarchy Design Methods

RepositoryManager.**repository_hierarchy_id**

Gets the hierarchy Id associated with this session.

Returns the hierarchy `Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`repository_hierarchy`**

Gets the hierarchy associated with this session.

Returns the hierarchy associated with this session

Return type `osid.hierarchy.Hierarchy`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`can_modify_repository_hierarchy`**`()`

Tests if this user can change the hierarchy.

A return of true does not guarantee successful authorization. A return of false indicates that it is known performing any update will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer these operations to an unauthorized user.

Returns `false` if changing this hierarchy is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`add_root_repository`**`(`*repository_id*`)`

Adds a root repository.

Parameters **`repository_id`** (`osid.id.Id`) – the `Id` of a repository

Raise `AlreadyExists` – `repository_id` is already in hierarchy

Raise `NotFound` – `repository_id` not found

Raise `NullArgument` – `repository_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`remove_root_repository`**`(`*repository_id*`)`

Removes a root repository.

Parameters **`repository_id`** (`osid.id.Id`) – the `Id` of a repository

Raise `NotFound` – `repository_id` not a root

Raise `NullArgument` – `repository_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`add_child_repository`**`(`*repository_id*, *child_id*`)`

Adds a child to a repository.

Parameters

- **repository_id** (`osid.id.Id`) – the `Id` of a repository
- **child_id** (`osid.id.Id`) – the `Id` of the new child

Raise `AlreadyExists` – `repository_id` is already a parent of `child_id`

Raise `NotFound` – `repository_id` or `child_id` not found

Raise `NullArgument` – `repository_id` or `child_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`remove_child_repository`**(*repository_id*, *child_id*)

Removes a child from a repository.

Parameters

- **repository_id** (`osid.id.Id`) – the `Id` of a repository
- **child_id** (`osid.id.Id`) – the `Id` of the new child

Raise `NotFound` – `repository_id` not a parent of `child_id`

Raise `NullArgument` – `repository_id` or `child_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`RepositoryManager.`**`remove_child_repositories`**(*repository_id*)

Removes all children from a repository.

Parameters **repository_id** (`osid.id.Id`) – the `Id` of a repository

Raise `NotFound` – `repository_id` not in hierarchy

Raise `NullArgument` – `repository_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Repository

Repository

class `dlkit.services.repository.`**`Repository`**(*provider_manager*, *catalog*, *runtime*, *proxy*, ***kwargs*)

Bases: *`dlkit.osid.objects.OsidCatalog`*, `dlkit.osid.sessions.OsidSession`

A repository defines a collection of assets.

`get_repository_record`(*repository_record_type*)

Gets the record corresponding to the given `Repository` record `Type`.

This method is used to retrieve an object implementing the requested record. The `repository_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(repository_record_type)` is `true` .

Parameters **`repository_record_type`** (`osid.type.Type`) – a repository record type

Returns the repository record

Return type `osid.repository.records.RepositoryRecord`

Raise `NullArgument` – `repository_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(repository_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Asset Lookup Methods

`Repository.`**`repository_id`**

Gets the `Repository Id` associated with this session.

Returns the `Repository Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Repository.`**`repository`**

Gets the `Repository` associated with this session.

Returns the `Repository` associated with this session

Return type `osid.repository.Repository`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`can_lookup_assets`**`()`

Tests if this user can perform `Asset` lookups.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations.

Returns `false` if lookup methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Repository.`**`use_comparative_asset_view`**`()`

The returns from the lookup methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`Repository.`**`use_plenary_asset_view`**`()`

A complete view of the `Asset` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`Repository.`**`use_federated_repository_view`**`()`

Federates the view for methods in this session.

A federated view will include compositions in repositories which are children of this repository in the repository hierarchy.

compliance: mandatory – This method is must be implemented.

`Repository.`**`use_isolated_repository_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts lookups to this repository only.

compliance: mandatory – This method is must be implemented.

`Repository.`**`get_asset`**`(`*asset_id*`)`

Gets the `Asset` specified by its `Id`.

In plenary mode, the exact `Id` is found or a `NotFound` results. Otherwise, the returned `Asset` may have a different `Id` than requested, such as the case where a duplicate `Id` was assigned to an `Asset` and retained for compatibility.

Parameters **`asset_id`** (`osid.id.Id`) – the `Id` of the `Asset` to retrieve

Returns the returned `Asset`

Return type `osid.repository.Asset`

Raise `NotFound` – no `Asset` found with the given `Id`

Raise `NullArgument` – `asset_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`get_assets_by_ids`**`(`*asset_ids*`)`

Gets an `AssetList` corresponding to the given `IdList`.

In plenary mode, the returned list contains all of the assets specified in the `Id` list, in the order of the list, including duplicates, or an error results if an `Id` in the supplied list is not found or inaccessible. Otherwise, inaccessible `Assets` may be omitted from the list and may present the elements in any order including returning a unique set.

Parameters **`asset_ids`** (`osid.id.IdList`) – the list of `Ids` to retrieve

Returns the returned `Asset list`

Return type `osid.repository.AssetList`

Raise `NotFound` – an `Id` was not found

Raise `NullArgument` – `asset_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`get_assets_by_genus_type`**(*asset_genus_type*)

Gets an `AssetList` corresponding to the given asset genus `Type` which does not include assets of types derived from the specified `Type`.

In plenary mode, the returned list contains all known assets or an error results. Otherwise, the returned list may contain only those assets that are accessible through this session.

Parameters **`asset_genus_type`** (`osid.type.Type`) – an asset genus type

Returns the returned `Asset list`

Return type `osid.repository.AssetList`

Raise `NullArgument` – `asset_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`get_assets_by_parent_genus_type`**(*asset_genus_type*)

Gets an `AssetList` corresponding to the given asset genus `Type` and include any additional assets with genus types derived from the specified `Type`.

In plenary mode, the returned list contains all known assets or an error results. Otherwise, the returned list may contain only those assets that are accessible through this session.

Parameters **`asset_genus_type`** (`osid.type.Type`) – an asset genus type

Returns the returned `Asset list`

Return type `osid.repository.AssetList`

Raise `NullArgument` – `asset_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`get_assets_by_record_type`**(*asset_record_type*)

Gets an `AssetList` containing the given asset record `Type`.

In plenary mode, the returned list contains all known assets or an error results. Otherwise, the returned list may contain only those assets that are accessible through this session.

Parameters **`asset_record_type`** (`osid.type.Type`) – an asset record type

Returns the returned `Asset list`

Return type `osid.repository.AssetList`

Raise `NullArgument` – `asset_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`get_assets_by_provider`**(*resource_id*)

Gets an `AssetList` from the given provider.

In plenary mode, the returned list contains all known assets or an error results. Otherwise, the returned list may contain only those assets that are accessible through this session.

Parameters **resource_id** (osid.id.Id) – a resource Id

Returns the returned Asset list

Return type osid.repository.AssetList

Raise NullArgument – resource_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Repository.**assets**

Gets all Assets.

In plenary mode, the returned list contains all known assets or an error results. Otherwise, the returned list may contain only those assets that are accessible through this session.

Returns a list of Assets

Return type osid.repository.AssetList

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Asset Query Methods

Repository.**repository_id**

Gets the Repository Id associated with this session.

Returns the Repository Id associated with this session

Return type osid.id.Id

compliance: mandatory – This method must be implemented.

Repository.**repository**

Gets the Repository associated with this session.

Returns the Repository associated with this session

Return type osid.repository.Repository

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Repository.**can_search_assets**()

Tests if this user can perform Asset searches.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a PermissionDenied. This is intended as a hint to an application that may opt not to offer search operations to unauthorized users.

Returns false if search methods are not authorized, true otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

`Repository.`**`use_federated_repository_view`**`()`

Federates the view for methods in this session.

A federated view will include compositions in repositories which are children of this repository in the repository hierarchy.

compliance: mandatory – This method is must be implemented.

`Repository.`**`use_isolated_repository_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts lookups to this repository only.

compliance: mandatory – This method is must be implemented.

`Repository.`**`asset_query`**

Gets an asset query.

Returns the asset query

Return type `osid.repository.AssetQuery`

compliance: mandatory – This method must be implemented.

`Repository.`**`get_assets_by_query`**`(`*asset_query*`)`

Gets a list of `Assets` matching the given asset query.

Parameters **`asset_query`** (`osid.repository.AssetQuery`) – the asset query

Returns the returned `AssetList`

Return type `osid.repository.AssetList`

Raise `NullArgument` – `asset_query` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – the `asset_query` is not of this service

compliance: mandatory – This method must be implemented.

Asset Search Methods

`Repository.`**`asset_search`**

Gets an asset search.

Returns the asset search

Return type `osid.repository.AssetSearch`

compliance: mandatory – This method must be implemented.

`Repository.`**`asset_search_order`**

Gets an asset search order.

The `AssetSearchOrder` is supplied to an `AssetSearch` to specify the ordering of results.

Returns the asset search order

Return type `osid.repository.AssetSearchOrder`

compliance: mandatory – This method must be implemented.

`Repository.`**`get_assets_by_search`**`(`*asset_query*, *asset_search*`)`

Gets the search results matching the given search query using the given search.

Parameters

- **asset_query** (osid.repository.AssetQuery) – the asset query
- **asset_search** (osid.repository.AssetSearch) – the asset search

Returns the asset search results

Return type osid.repository.AssetSearchResults

Raise NullArgument – asset_query or asset_search is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

Raise Unsupported – asset_query or asset_search is not of this service

compliance: mandatory – This method must be implemented.

Repository.**get_asset_query_from_inspector**(*asset_query_inspector*)

Gets an asset query from an inspector.

The inspector is available from a AssetSearchResults.

Parameters **asset_query_inspector** (osid.repository.AssetQueryInspector) – an asset query inspector

Returns the asset query

Return type osid.repository.AssetQuery

Raise NullArgument – asset_query_inspector is null

Raise Unsupported – asset_query_inspector is not of this service

compliance: mandatory – This method must be implemented.

Asset Admin Methods

Repository.**repository_id**

Gets the Repository Id associated with this session.

Returns the Repository Id associated with this session

Return type osid.id.Id

compliance: mandatory – This method must be implemented.

Repository.**repository**

Gets the Repository associated with this session.

Returns the Repository associated with this session

Return type osid.repository.Repository

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Repository.**can_create_assets**()

Tests if this user can create Assets.

A return of true does not guarantee successful authorization. A return of false indicates that it is known creating an `Asset` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer create operations to an unauthorized user.

Returns `false` if `Asset` creation is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Repository.`**`can_create_asset_with_record_types`**(*asset_record_types*)

Tests if this user can create a single `Asset` using the desired record types.

While `RepositoryManager.getAssetRecordTypes()` can be used to examine which records are supported, this method tests which record(s) are required for creating a specific `Asset`. Providing an empty array tests if an `Asset` can be created with no records.

Parameters **`asset_record_types`** (`osid.type.Type[]`) – array of asset record types

Returns `true` if `Asset` creation using the specified record `Types` is supported, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `asset_record_types` is `null`

compliance: mandatory – This method must be implemented.

`Repository.`**`get_asset_form_for_create`**(*asset_record_types*)

Gets the asset form for creating new assets.

A new form should be requested for each create transaction.

Parameters **`asset_record_types`** (`osid.type.Type[]`) – array of asset record types

Returns the asset form

Return type `osid.repository.AssetForm`

Raise `NullArgument` – `asset_record_types` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – unable to get form for requested record types

compliance: mandatory – This method must be implemented.

`Repository.`**`create_asset`**(*asset_form*)

Creates a new `Asset`.

Parameters **`asset_form`** (`osid.repository.AssetForm`) – the form for this `Asset`

Returns the new `Asset`

Return type `osid.repository.Asset`

Raise `IllegalState` – `asset_form` already used in a create transaction

Raise `InvalidArgument` – one or more of the form elements is invalid

Raise `NullArgument` – `asset_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `asset_form` did not originate from `get_asset_form_for_create()`

compliance: mandatory – This method must be implemented.

`Repository.`**`can_update_assets`**`()`

Tests if this user can update `Assets`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating an `Asset` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer update operations to an unauthorized user.

Returns `false` if `Asset` modification is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Repository.`**`get_asset_form_for_update`**`(`*asset_id*`)`

Gets the asset form for updating an existing asset.

A new asset form should be requested for each update transaction.

Parameters **`asset_id`** (`osid.id.Id`) – the `Id` of the `Asset`

Returns the asset form

Return type `osid.repository.AssetForm`

Raise `NotFound` – `asset_id` is not found

Raise `NullArgument` – `asset_id` is null

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`update_asset`**`(`*asset_form*`)`

Updates an existing asset.

Parameters **`asset_form`** (`osid.repository.AssetForm`) – the form containing the elements to be updated

Raise `IllegalState` – `asset_form` already used in anupdate transaction

Raise `InvalidArgument` – the form contains an invalid value

Raise `NullArgument` – `asset_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `asset_form` did not originate from `get_asset_form_for_update()`

compliance: mandatory – This method must be implemented.

`Repository.`**`can_delete_assets`**`()`

Tests if this user can delete `Assets`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting an `Asset` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer delete operations to an unauthorized user.

Returns `false` if `Asset` deletion is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Repository.`**`delete_asset`**`(`*asset_id*`)`

Deletes an `Asset`.

Parameters **`asset_id`** (`osid.id.Id`) – the `Id` of the `Asset` to remove

Raise `NotFound` – `asset_id` not found

Raise `NullArgument` – `asset_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`can_manage_asset_aliases`**`()`

Tests if this user can manage `Id` aliases for `Assets`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known changing an alias will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer alias operations to an unauthorized user.

Returns `false` if `Asset` aliasing is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Repository.`**`alias_asset`**`(`*asset_id*, *alias_id*`)`

Adds an `Id` to an `Asset` for the purpose of creating compatibility.

The primary `Id` of the `Asset` is determined by the provider. The new `Id` performs as an alias to the primary `Id`. If the alias is a pointer to another asset, it is reassigned to the given asset `Id`.

Parameters

- **`asset_id`** (`osid.id.Id`) – the `Id` of an `Asset`
- **`alias_id`** (`osid.id.Id`) – the alias `Id`

Raise `AlreadyExists` – `alias_id` is already assigned

Raise `NotFound` – `asset_id` not found

Raise `NullArgument` – `asset_id` or `alias_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`can_create_asset_content`**`()`

Tests if this user can create content for `Assets`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known creating an `AssetContent` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer create operations to an unauthorized user.

Returns `false` if `Asset` content creation is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Repository.`**`can_create_asset_content_with_record_types`**(*asset_content_record_types*)

Tests if this user can create an `AssetContent` using the desired record types.

While `RepositoryManager.getAssetContentRecordTypes()` can be used to test which records are supported, this method tests which records are required for creating a specific `AssetContent`. Providing an empty array tests if an `AssetContent` can be created with no records.

Parameters **`asset_content_record_types`** (`osid.type.Type[]`) – array of asset content record types

Returns `true` if `AssetContent` creation using the specified `Types` is supported, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `asset_content_record_types` is `null`

compliance: mandatory – This method must be implemented.

`Repository.`**`get_asset_content_form_for_create`**(*asset_id*, *asset_content_record_types*)

Gets an asset content form for creating new assets.

Parameters

- **`asset_id`** (`osid.id.Id`) – the `Id` of an `Asset`
- **`asset_content_record_types`** (`osid.type.Type[]`) – array of asset content record types

Returns the asset content form

Return type `osid.repository.AssetContentForm`

Raise `NotFound` – `asset_id` is not found

Raise `NullArgument` – `asset_id` or `asset_content_record_types` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – unable to get form for requested record types

compliance: mandatory – This method must be implemented.

`Repository.`**`create_asset_content`**(*asset_content_form*)

Creates new `AssetContent` for a given asset.

Parameters **`asset_content_form`** (`osid.repository.AssetContentForm`) – the form for this `AssetContent`

Returns the new `AssetContent`

Return type `osid.repository.AssetContent`

Raise `IllegalState` – `asset_content_form` already used in a create transaction

Raise `InvalidArgument` – one or more of the form elements is invalid

Raise `NullArgument` – `asset_content_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `asset_content_form` did not originate from `get_asset_content_form_for_create()`

compliance: mandatory – This method must be implemented.

`Repository.`**`can_update_asset_contents`**`()`

Tests if this user can update `AssetContent`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating an `AssetContent` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer update operations to an unauthorized user.

Returns `false` if `AssetContent` modification is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Repository.`**`get_asset_content_form_for_update`**`(`*asset_content_id*`)`

Gets the asset content form for updating an existing asset content.

A new asset content form should be requested for each update transaction.

Parameters **`asset_content_id`** (`osid.id.Id`) – the `Id` of the `AssetContent`

Returns the asset content form

Return type `osid.repository.AssetContentForm`

Raise `NotFound` – `asset_content_id` is not found

Raise `NullArgument` – `asset_content_id` is `null`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`Repository.`**`update_asset_content`**`(`*asset_content_form*`)`

Updates an existing asset content.

Parameters **`asset_content_form`** (`osid.repository.AssetContentForm`) – the form containing the elements to be updated

Raise `IllegalState` – `asset_content_form` already used in an update transaction

Raise `InvalidArgument` – the form contains an invalid value

Raise `NullArgument` – `asset_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `asset_content_form` did not originate from `get_asset_content_form_for_update()`

compliance: mandatory – This method must be implemented.

`Repository.`**`can_delete_asset_contents`**`()`

Tests if this user can delete `AssetsContents`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting an `AssetContent` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer delete operations to an unauthorized user.

Returns `false` if `AssetContent` deletion is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Repository.`**`delete_asset_content`**(*asset_content_id*)

Deletes content from an `Asset`.

Parameters **`asset_content_id`** (`osid.id.Id`) – the `Id` of the `AssetContent`

Raise `NotFound` – `asset_content_id` is not found

Raise `NullArgument` – `asset_content_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Asset Notification Methods

`Repository.`**`repository_id`**

Gets the `Repository Id` associated with this session.

Returns the `Repository Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Repository.`**`repository`**

Gets the `Repository` associated with this session.

Returns the `Repository` associated with this session

Return type `osid.repository.Repository`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`can_register_for_asset_notifications`**()

Tests if this user can register for `Asset` notifications.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer notification operations.

Returns `false` if notification methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Repository.`**`use_federated_repository_view`**()

Federates the view for methods in this session.

A federated view will include compositions in repositories which are children of this repository in the repository hierarchy.

compliance: mandatory – This method is must be implemented.

Repository.**use_isolated_repository_view**()

Isolates the view for methods in this session.

An isolated view restricts lookups to this repository only.

compliance: mandatory – This method is must be implemented.

Repository.**register_for_new_assets**()

Register for notifications of new assets.

`AssetReceiver.newAssets()` is invoked when a new `Asset` appears in this repository.

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Repository.**register_for_new_assets_by_genus_type**(*asset_genus_type*)

Registers for notification of new assets of the given asset genus type.

`AssetReceiver.newAssets()` is invoked when an asset is appears in this repository.

Parameters **asset_genus_type** (`osid.type.Type`) – the genus type of the `Asset` to monitor

Raise `NullArgument` – `asset_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Repository.**register_for_changed_assets**()

Registers for notification of updated assets.

`AssetReceiver.changedAssets()` is invoked when an asset in this repository is changed.

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Repository.**register_for_changed_assets_by_genus_type**(*asset_genus_type*)

Registers for notification of updated assets of the given asset genus type.

`AssetReceiver.changedAssets()` is invoked when an asset in this repository is changed.

Parameters **asset_genus_type** (`osid.type.Type`) – the genus type of the `Asset` to monitor

Raise `NullArgument` – `asset_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Repository.**register_for_changed_asset**(*asset_id*)

Registers for notification of an updated asset.

`AssetReceiver.changedAssets()` is invoked when the specified asset in this repository is changed.

Parameters **asset_id** (`osid.id.Id`) – the `Id` of the `Asset` to monitor

Raise `NullArgument` – `asset_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`register_for_deleted_assets`**`()`

Registers for notification of deleted assets.

`AssetReceiver.deletedAssets()` is invoked when an asset is deleted or removed from this repository.

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`register_for_deleted_assets_by_genus_type`**`(`*asset_genus_type*`)`

Registers for notification of deleted assets of the given asset genus type.

`AssetReceiver.deletedAssets()` is invoked when an asset is deleted or removed from this repository.

Parameters **`asset_genus_type`** (`osid.type.Type`) – the genus type of the `Asset` to monitor

Raise `NullArgument` – `asset_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`register_for_deleted_asset`**`(`*asset_id*`)`

Registers for notification of a deleted asset.

`AssetReceiver.deletedAssets()` is invoked when the specified asset is deleted or removed from this repository.

Parameters **`asset_id`** (`osid.id.Id`) – the `Id` of the `Asset` to monitor

Raise `NullArgument` – `asset_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`reliable_asset_notifications`**`()`

Reliable notifications are desired.

In reliable mode, notifications are to be acknowledged using `acknowledge_item_notification()` .

compliance: mandatory – This method is must be implemented.

`Repository.`**`unreliable_asset_notifications`**`()`

Unreliable notifications are desired.

In unreliable mode, notifications do not need to be acknowledged.

compliance: mandatory – This method is must be implemented.

`Repository.`**`acknowledge_asset_notification`**`(`*notification_id*`)`
Acknowledge an asset notification.

Parameters **`notification_id`** (`osid.id.Id`) – the `Id` of the notification

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Asset Repository Methods

`Repository.`**`can_lookup_asset_repository_mappings`**`()`
Tests if this user can perform lookups of asset/repository mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known lookup methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if looking up mappings is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Repository.`**`use_comparative_repository_view`**`()`
The returns from the lookup methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`Repository.`**`use_plenary_repository_view`**`()`
A complete view of the `Asset` and `Repository` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`Repository.`**`get_asset_ids_by_repository`**`(`*repository_id*`)`
Gets the list of `Asset Ids` associated with a `Repository`.

Parameters **`repository_id`** (`osid.id.Id`) – `Id` of the `Repository`

Returns list of related asset `Ids`

Return type `osid.id.IdList`

Raise `NotFound` – `repository_id` is not found

Raise `NullArgument` – `repository_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`get_assets_by_repository`**`(`*repository_id*`)`
Gets the list of `Assets` associated with a `Repository`.

Parameters **`repository_id`** (`osid.id.Id`) – `Id` of the `Repository`

Returns list of related assets

Return type `osid.repository.AssetList`

Raise `NotFound` – `repository_id` is not found

Raise `NullArgument` – `repository_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`get_asset_ids_by_repositories`**`(`*repository_ids*`)`

Gets the list of `Asset Ids` corresponding to a list of `Repository` objects.

Parameters **`repository_ids`** (`osid.id.IdList`) – list of repository `Ids`

Returns list of asset `Ids`

Return type `osid.id.IdList`

Raise `NullArgument` – `repository_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`get_assets_by_repositories`**`(`*repository_ids*`)`

Gets the list of `Assets` corresponding to a list of `Repository` objects.

Parameters **`repository_ids`** (`osid.id.IdList`) – list of repository `Ids`

Returns list of assets

Return type `osid.repository.AssetList`

Raise `NullArgument` – `repository_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`get_repository_ids_by_asset`**`(`*asset_id*`)`

Gets the list of `Repository Ids` mapped to an `Asset`.

Parameters **`asset_id`** (`osid.id.Id`) – `Id` of an `Asset`

Returns list of repository `Ids`

Return type `osid.id.IdList`

Raise `NotFound` – `asset_id` is not found

Raise `NullArgument` – `asset_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`get_repositories_by_asset`**`(`*asset_id*`)`

Gets the list of `Repository` objects mapped to an `Asset`.

Parameters **`asset_id`** (`osid.id.Id`) – `Id` of an `Asset`

Returns list of repositories

Return type `osid.repository.RepositoryList`

Raise `NotFound` – `asset_id` is not found

Raise `NullArgument` – `asset_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Asset Repository Assignment Methods

`Repository.`**`can_assign_assets`**`()`

Tests if this user can alter asset/repository mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known mapping methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer assignment operations to unauthorized users.

Returns `false` if mapping is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Repository.`**`can_assign_assets_to_repository`**`(`*repository_id*`)`

Tests if this user can alter asset/repository mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known mapping methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer assignment operations to unauthorized users.

Parameters **`repository_id`** (`osid.id.Id`) – the `Id` of the `Repository`

Returns `false` if mapping is not authorized, `true` otherwise

Return type `boolean`

Raise `NullArgument` – `repository_id` is `null`

compliance: mandatory – This method must be implemented.

`Repository.`**`get_assignable_repository_ids`**`(`*repository_id*`)`

Gets a list of repositories including and under the given repository node in which any composition can be assigned.

Parameters **`repository_id`** (`osid.id.Id`) – the `Id` of the `Repository`

Returns list of assignable repository `Ids`

Return type `osid.id.IdList`

Raise `NullArgument` – `repository_id` is `null`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`Repository.`**`get_assignable_repository_ids_for_asset`**(*repository_id*, *asset_id*)

Gets a list of repositories including and under the given repository node in which a specific asset can be assigned.

Parameters

- **`repository_id`** (`osid.id.Id`) – the `Id` of the `Repository`
- **`asset_id`** (`osid.id.Id`) – the `Id` of the `Asset`

Returns list of assignable repository `Ids`

Return type `osid.id.IdList`

Raise `NullArgument` – `repository_id` or `asset_id` is `null`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`Repository.`**`assign_asset_to_repository`**(*asset_id*, *repository_id*)

Adds an existing `Asset` to a `Repository`.

Parameters

- **`asset_id`** (`osid.id.Id`) – the `Id` of the `Asset`
- **`repository_id`** (`osid.id.Id`) – the `Id` of the `Repository`

Raise `AlreadyExists` – `asset_id` already assigned to `repository_id`

Raise `NotFound` – `asset_id` or `repository_id` not found

Raise `NullArgument` – `asset_id` or `repository_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`unassign_asset_from_repository`**(*asset_id*, *repository_id*)

Removes an `Asset` from a `Repository`.

Parameters

- **`asset_id`** (`osid.id.Id`) – the `Id` of the `Asset`
- **`repository_id`** (`osid.id.Id`) – the `Id` of the `Repository`

Raise `NotFound` – `asset_id` or `repository_id` not found or `asset_id` not assigned to `repository_id`

Raise `NullArgument` – `asset_id` or `repository_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Asset Composition Methods

`Repository.`**`repository_id`**

Gets the `Repository Id` associated with this session.

Returns the `Repository Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Repository.`**`repository`**

Gets the `Repository` associated with this session.

Returns the `Repository` associated with this session

Return type `osid.repository.Repository`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`can_access_asset_compositions`**`()`

Tests if this user can perform composition lookups.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if lookup methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Repository.`**`use_comparative_asset_composition_view`**`()`

The returns from the lookup methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`Repository.`**`use_plenary_asset_composition_view`**`()`

A complete view of the returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`Repository.`**`use_federated_repository_view`**`()`

Federates the view for methods in this session.

A federated view will include compositions in repositories which are children of this repository in the repository hierarchy.

compliance: mandatory – This method is must be implemented.

`Repository.`**`use_isolated_repository_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts lookups to this repository only.

compliance: mandatory – This method is must be implemented.

`Repository.`**`get_composition_assets`**`(`*composition_id*`)`

Gets the list of assets mapped to the given `Composition`.

Parameters **`composition_id`** (`osid.id.Id`) – `Id` of the `Composition`

Returns list of assets

Return type `osid.repository.AssetList`

Raise `NotFound` – `composition_id` not found

Raise `NullArgument` – `composition_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method is must be implemented.

`Repository.`**`get_compositions_by_asset`**`(`*asset_id*`)`

Gets a list of compositions including the given asset.

Parameters **`asset_id`** (`osid.id.Id`) – `Id` of the `Asset`

Returns the returned `Composition list`

Return type `osid.repository.CompositionList`

Raise `NotFound` – `asset_id` is not found

Raise `NullArgument` – `asset_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Asset Composition Design Methods

`Repository.`**`repository_id`**

Gets the `Repository Id` associated with this session.

Returns the `Repository Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Repository.`**`repository`**

Gets the `Repository` associated with this session.

Returns the `Repository` associated with this session

Return type `osid.repository.Repository`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`can_compose_assets`**`()`

Tests if this user can manage mapping of `Assets` to `Compositions`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as an application hint that may opt not to offer composition operations.

Returns `false` if asset composiion is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Repository.`**`add_asset`**`(`*asset_id*, *composition_id*`)`

Appends an asset to a composition.

Parameters

- **`asset_id`** (`osid.id.Id`) – `Id` of the `Asset`
- **`composition_id`** (`osid.id.Id`) – `Id` of the `Composition`

Raise `AlreadyExists` – `asset_id` already part `composition_id`

Raise `NotFound` – `asset_id` or `composition_id` not found

Raise `NullArgument` – `asset_id` or `composition_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization fauilure

compliance: mandatory – This method must be implemented.

`Repository.`**`move_asset_ahead`**`(`*asset_id*, *composition_id*, *reference_id*`)`

Reorders assets in a composition by moving the specified asset in front of a reference asset.

Parameters

- **`asset_id`** (`osid.id.Id`) – `Id` of the `Asset`
- **`composition_id`** (`osid.id.Id`) – `Id` of the `Composition`
- **`reference_id`** (`osid.id.Id`) – `Id` of the reference `Asset`

Raise `NotFound` – `asset_id` or `reference_id not found in composition_id`

Raise `NullArgument – asset_id, reference_id` or `composition_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization fauilure

compliance: mandatory – This method must be implemented.

`Repository.`**`move_asset_behind`**`(`*asset_id*, *composition_id*, *reference_id*`)`

Reorders assets in a composition by moving the specified asset behind of a reference asset.

Parameters

- **`asset_id`** (`osid.id.Id`) – `Id` of the `Asset`
- **`composition_id`** (`osid.id.Id`) – `Id` of the `Composition`
- **`reference_id`** (`osid.id.Id`) – `Id` of the reference `Asset`

Raise `NotFound` – `asset_id` or `reference_id not found in composition_id`

Raise `NullArgument – asset_id, reference_id` or `composition_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization fauilure

compliance: mandatory – This method must be implemented.

`Repository.`**`order_assets`**(*asset_ids*, *composition_id*)

Reorders a set of assets in a composition.

Parameters

- **`asset_ids`** (`osid.id.Id[]`) – `Ids` for a set of `Assets`
- **`composition_id`** (`osid.id.Id`) – `Id` of the `Composition`

Raise `NotFound` – `composition_id` not found or, an `asset_id` not related to `composition_id`

Raise `NullArgument` – `instruction_ids` or `agenda_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`remove_asset`**(*asset_id*, *composition_id*)

Removes an `Asset` from a `Composition`.

Parameters

- **`asset_id`** (`osid.id.Id`) – `Id` of the `Asset`
- **`composition_id`** (`osid.id.Id`) – `Id` of the `Composition`

Raise `NotFound` – `asset_id not found in composition_id`

Raise `NullArgument` – `asset_id` or `composition_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization fauilure

compliance: mandatory – This method must be implemented.

Composition Lookup Methods

`Repository.`**`repository_id`**

Gets the `Repository Id` associated with this session.

Returns the `Repository Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Repository.`**`repository`**

Gets the `Repository` associated with this session.

Returns the `Repository` associated with this session

Return type `osid.repository.Repository`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`can_lookup_compositions`**()

Tests if this user can perform `Composition` lookups.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if lookup methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Repository.`**`use_comparative_composition_view`**`()`

The returns from the lookup methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`Repository.`**`use_plenary_composition_view`**`()`

A complete view of the `Composition` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`Repository.`**`use_federated_repository_view`**`()`

Federates the view for methods in this session.

A federated view will include compositions in repositories which are children of this repository in the repository hierarchy.

compliance: mandatory – This method is must be implemented.

`Repository.`**`use_isolated_repository_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts lookups to this repository only.

compliance: mandatory – This method is must be implemented.

`Repository.`**`use_active_composition_view`**`()`

Only active compositions are returned by methods in this session.

compliance: mandatory – This method is must be implemented.

`Repository.`**`use_any_status_composition_view`**`()`

All active and inactive compositions are returned by methods in this session.

compliance: mandatory – This method is must be implemented.

`Repository.`**`use_sequestered_composition_view`**`()`

The methods in this session omit sequestered compositions.

compliance: mandatory – This method is must be implemented.

`Repository.`**`use_unsequestered_composition_view`**`()`

The methods in this session return all compositions, including sequestered compositions.

compliance: mandatory – This method is must be implemented.

`Repository.`**`get_composition`**`(`*composition_id*`)`

Gets the `Composition` specified by its `Id`.

Parameters **`composition_id`** (`osid.id.Id`) – `Id` of the `Composiiton`

Returns the composition

Return type `osid.repository.Composition`

Raise `NotFound` – `composition_id` not found

Raise `NullArgument` – `composition_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method is must be implemented.

`Repository.`**`get_compositions_by_ids`**(*composition_ids*)

Gets a `CompositionList` corresponding to the given `IdList`.

Parameters **`composition_ids`** (`osid.id.IdList`) – the list of `Ids` to retrieve

Returns the returned `Composition list`

Return type `osid.repository.CompositionList`

Raise `NotFound` – an `Id` was not found

Raise `NullArgument` – `composition_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`get_compositions_by_genus_type`**(*composition_genus_type*)

Gets a `CompositionList` corresponding to the given composition genus `Type` which does not include compositions of types derived from the specified `Type`.

Parameters **`composition_genus_type`** (`osid.type.Type`) – a composition genus type

Returns the returned `Composition list`

Return type `osid.repository.CompositionList`

Raise `NullArgument` – `composition_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`get_compositions_by_parent_genus_type`**(*composition_genus_type*)

Gets a `CompositionList` corresponding to the given composition genus `Type` and include any additional compositions with genus types derived from the specified `Type`.

Parameters **`composition_genus_type`** (`osid.type.Type`) – a composition genus type

Returns the returned `Composition list`

Return type `osid.repository.CompositionList`

Raise `NullArgument` – `composition_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`get_compositions_by_record_type`**(*composition_record_type*)

Gets a `CompositionList` containing the given composition record `Type`.

Parameters **`composition_record_type`** (`osid.type.Type`) – a composition record type

Returns the returned `Composition list`

Return type `osid.repository.CompositionList`

Raise `NullArgument` – `composition_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`get_compositions_by_provider`**(*resource_id*)

Gets a `CompositionList` from the given provider ````.

In plenary mode, the returned list contains all known compositions or an error results. Otherwise, the returned list may contain only those compositions that are accessible through this session.

In sequestered mode, no sequestered compositions are returned. In unsequestered mode, all compositions are returned.

Parameters **`resource_id`** (`osid.id.Id`) – a resource `Id`

Returns the returned `Composition list`

Return type `osid.repository.CompositionList`

Raise `NullArgument` – `resource_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`compositions`**

Gets all `Compositions`.

Returns a list of `Compositions`

Return type `osid.repository.CompositionList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Composition Query Methods

`Repository.`**`repository_id`**

Gets the `Repository Id` associated with this session.

Returns the `Repository Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Repository.`**`repository`**

Gets the `Repository` associated with this session.

Returns the `Repository` associated with this session

Return type `osid.repository.Repository`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`can_search_compositions`**`()`

Tests if this user can perform `Composition` searches.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer search operations to unauthorized users.

Returns `false` if search methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Repository.`**`use_federated_repository_view`**`()`

Federates the view for methods in this session.

A federated view will include compositions in repositories which are children of this repository in the repository hierarchy.

compliance: mandatory – This method is must be implemented.

`Repository.`**`use_isolated_repository_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts lookups to this repository only.

compliance: mandatory – This method is must be implemented.

`Repository.`**`use_sequestered_composition_view`**`()`

The methods in this session omit sequestered compositions.

compliance: mandatory – This method is must be implemented.

`Repository.`**`use_unsequestered_composition_view`**`()`

The methods in this session return all compositions, including sequestered compositions.

compliance: mandatory – This method is must be implemented.

`Repository.`**`composition_query`**

Gets a composition query.

Returns the composition query

Return type `osid.repository.CompositionQuery`

compliance: mandatory – This method must be implemented.

`Repository.`**`get_compositions_by_query`**`(`*composition_query*`)`

Gets a list of `Compositions` matching the given composition query.

Parameters **`composition_query`** (`osid.repository.CompositionQuery`) – the composition query

Returns the returned `CompositionList`

Return type `osid.repository.CompositionList`

Raise `NullArgument` – `composition_query` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `composition_query` is not of this service

compliance: mandatory – This method must be implemented.

Composition Search Methods

`Repository.`**`composition_search`**

Gets a composition search.

Returns the composition search

Return type `osid.repository.CompositionSearch`

compliance: mandatory – This method must be implemented.

`Repository.`**`composition_search_order`**

Gets a composition search order.

The `CompositionSearchOrder` is supplied to an `CompositionSearch` to specify the ordering of results.

Returns the composition search order

Return type `osid.repository.CompositionSearchOrder`

compliance: mandatory – This method must be implemented.

`Repository.`**`get_compositions_by_search`**(*composition_query*, *composition_search*)

Gets the search results matching the given search query using the given search.

Parameters

- **`composition_query`** (`osid.repository.CompositionQuery`) – the composition query
- **`composition_search`** (`osid.repository.CompositionSearch`) – the composition search

Returns the composition search results

Return type `osid.repository.CompositionSearchResults`

Raise `NullArgument` – `composition_query` or `composition_search` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `composition_query` or `composition_search` is not of this service

compliance: mandatory – This method must be implemented.

`Repository.`**`get_composition_query_from_inspector`**(*composition_query_inspector*)

Gets a composition query from an inspector.

The inspector is available from a `CompositionSearchResults`.

Parameters **`composition_query_inspector`** (`osid.repository.CompositionQueryInspector`) – a composition query inspector

Returns the composition query

Return type `osid.repository.CompositionQuery`

Raise `NullArgument` – `composition_query_inspector` is `null`

Raise `Unsupported` – `composition_query_inspector` is not of this service

compliance: mandatory – This method must be implemented.

Composition Admin Methods

`Repository.`**`repository_id`**

Gets the `Repository Id` associated with this session.

Returns the `Repository Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Repository.`**`repository`**

Gets the `Repository` associated with this session.

Returns the `Repository` associated with this session

Return type `osid.repository.Repository`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`can_create_compositions`**`()`

Tests if this user can create `Compositions`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known creating a `Composition` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer create operations to unauthorized users.

Returns `false` if `Composition` creation is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Repository.`**`can_create_composition_with_record_types`**`(`*composition_record_types*`)`

Tests if this user can create a single `Composition` using the desired record types.

While `RepositoryManager.getCompositionRecordTypes()` can be used to examine which records are supported, this method tests which record(s) are required for creating a specific `Composition`. Providing an empty array tests if a `Composition` can be created with no records.

Parameters **`composition_record_types`** (`osid.type.Type[]`) – array of composition record types

Returns `true` if `Composition` creation using the specified `Types` is supported, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `composition_record_types` is `null`

compliance: mandatory – This method must be implemented.

`Repository.`**`get_composition_form_for_create`**`(`*composition_record_types*`)`

Gets the composition form for creating new compositions.

A new form should be requested for each create transaction.

Parameters **`composition_record_types`** (`osid.type.Type[]`) – array of composition record types

Returns the composition form

Return type `osid.repository.CompositionForm`

Raise `NullArgument` – `composition_record_types` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – unable to get form for requested record types

compliance: mandatory – This method must be implemented.

`Repository.`**`create_composition`**`(`*composiiton_form*`)`

Creates a new `Composition`.

Parameters **`composiiton_form`** (`osid.repository.CompositionForm`) – the form for this `Composition`

Returns the new `Composition`

Return type `osid.repository.Composition`

Raise `IllegalState` – `composition_form` already used in a create transaction

Raise `InvalidArgument` – one or more of the form elements is invalid

Raise `NullArgument` – `composition_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `composition_form` did not originate from `get_composition_form_for_create()`

compliance: mandatory – This method must be implemented.

`Repository.`**`can_update_compositions`**`()`

Tests if this user can update `Compositions`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating a `Composition` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer update operations to unauthorized users.

Returns `false` if `Composition` modification is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Repository.`**`get_composition_form_for_update`**`(`*composition_id*`)`

Gets the composition form for updating an existing composition.

A new composition form should be requested for each update transaction.

Parameters **`composition_id`** (`osid.id.Id`) – the `Id` of the `Composition`

Returns the composition form

Return type `osid.repository.CompositionForm`

Raise `NotFound` – `composition_id` is not found

Raise `NullArgument` – `composition_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`update_composition`**`(`*compositon_form*`)`

Updates an existing composition.

Parameters **`composiiton_form`** (`osid.repository.CompositionForm`) – the form containing the elements to be updated

Raise `IllegalState` – `composition_form` already used in an update transaction

Raise `InvalidArgument` – the form contains an invalid value

Raise `NullArgument` – `composition_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `composition_form` did not originate from `get_composition_form_for_update()`

compliance: mandatory – This method must be implemented.

`Repository.`**`can_delete_compositions`**`()`

Tests if this user can delete `Compositions`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting a `Composition` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer delete operations to unauthorized users.

Returns `false` if `Composition` deletion is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Repository.`**`delete_composition`**`(`*composition_id*`)`

Deletes a `Composition`.

Parameters **`composition_id`** (`osid.id.Id`) – the `Id` of the `Composition` to remove

Raise `NotFound` – `composition_id` not found

Raise `NullArgument` – `composition_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`delete_composition_node`**`(`*composition_id*`)`

Deletes a `Composition` and all contained children.

Parameters **`composition_id`** (`osid.id.Id`) – the `Id` of the `Composition` to remove

Raise `NotFound` – `composition_id` not found

Raise `NullArgument` – `composition_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`add_composition_child`**`(`*composition_id*, *child_composition_id*`)`

Adds a composition to a parent composition.

Parameters

- **`composition_id`** (`osid.id.Id`) – the `Id` of a parent `Composition`
- **`child_composition_id`** (`osid.id.Id`) – the `Id` of a child `Composition`

Raise `AlreadyExists` – `child_composition_id` is already a child of `composition_id`

Raise `NotFound` – `composition_id` or `child_composition_id` is not found

Raise `NullArgument` – `composition_id` or `child_composition_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`remove_composition_child`**`(`*composition_id*, *child_composition_id*`)`

Removes a composition from a parent composition.

Parameters

- **`composition_id`** (`osid.id.Id`) – the `Id` of a parent `Composition`
- **`child_composition_id`** (`osid.id.Id`) – the `Id` of a child `Composition`

Raise `NotFound` – `composition_id` or `child_composition_id` is not found or not related

Raise `NullArgument` – `composition_id` or `child_composition_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`can_manage_composition_aliases`**`()`

Tests if this user can manage `Id` aliases for `Compositions`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known changing an alias will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer alias operations to an unauthorized user.

Returns `false` if `Composition` aliasing is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Repository.`**`alias_composition`**`(`*composition_id*, *alias_id*`)`

Adds an `Id` to a `Composition` for the purpose of creating compatibility.

The primary `Id` of the `Composition` is determined by the provider. The new `Id` is an alias to the primary `Id`. If the alias is a pointer to another composition, it is reassigned to the given composition `Id`.

Parameters

- **`composition_id`** (`osid.id.Id`) – the `Id` of a `Composition`
- **`alias_id`** (`osid.id.Id`) – the alias `Id`

Raise `AlreadyExists` – `alias_id` is in use as a primary `Id`

Raise `NotFound` – `composition_id` not found

Raise `NullArgument` – `composition_id` or `alias_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Composition Repository Methods

`Repository.`**`use_comparative_composition_repository_view`**`()`

The returns from the lookup methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`Repository.`**`use_plenary_composition_repository_view`**`()`

A complete view of the `Composition` and `Repository` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`Repository.`**`can_lookup_composition_repository_mappings`**`()`

Tests if this user can perform lookups of composition/repository mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known lookup methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if looking up mappings is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Repository.`**`get_composition_ids_by_repository`**`(`*repository_id*`)`

Gets the list of `Composition Ids` associated with a `Repository`.

Parameters **`repository_id`** (`osid.id.Id`) – `Id` of the `Repository`

Returns list of related composition `Ids`

Return type `osid.id.IdList`

Raise `NotFound` – `repository_id` is not found

Raise `NullArgument` – `repository_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`get_compositions_by_repository`**`(`*repository_id*`)`

Gets the list of `Compositions` associated with a `Repository`.

Parameters **`repository_id`** (`osid.id.Id`) – `Id` of the `Repository`

Returns list of related compositions

Return type `osid.repository.CompositionList`

Raise `NotFound` – `repository_id` is not found

Raise `NullArgument` – `repository_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`get_composition_ids_by_repositories`**`(`*repository_ids*`)`

Gets the list of `Composition Ids` corresponding to a list of `Repository` objects.

Parameters **`repository_ids`** (`osid.id.IdList`) – list of repository `Ids`

Returns list of composition `Ids`

Return type `osid.id.IdList`

Raise `NullArgument` – `repository_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`get_compoitions_by_repositories`**`(`*repository_ids*`)`

Gets the list of `Compositions` corresponding to a list of `Repository` objects.

Parameters **`repository_ids`** (`osid.id.IdList`) – list of repository `Ids`

Returns list of Compositions

Return type `osid.repository.CompositionList`

Raise `NullArgument` – `repository_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`get_repository_ids_by_composition`**`(`*composition_id*`)`

Gets the `Repository Ids` mapped to a `Composition`.

Parameters **`composition_id`** (`osid.id.Id`) – `Id` of a `Composition`

Returns list of repository `Ids`

Return type `osid.id.IdList`

Raise `NotFound` – `composition_id` is not found

Raise `NullArgument` – `composition_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`get_repositories_by_composition`**`(`*composition_id*`)`

Gets the `Repository` objects mapped to a `Composition`.

Parameters **`composition_id`** (`osid.id.Id`) – `Id` of a `Composition`

Returns list of repositories

Return type `osid.repository.RepositoryList`

Raise `NotFound` – `composition_id` is not found

Raise `NullArgument` – `composition_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Composition Repository Assignment Methods

`Repository.`**`can_assign_compositions`**`()`

Tests if this user can alter composition/repository mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known mapping methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer assignment operations to unauthorized users.

Returns `false` if mapping is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Repository.`**`can_assign_compositions_to_repository`**`(`*repository_id*`)`

Tests if this user can alter composition/repository mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known mapping methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer assignment operations to unauthorized users.

Parameters **`repository_id`** (`osid.id.Id`) – the `Id` of the `Repository`

Returns `false` if mapping is not authorized, `true` otherwise

Return type `boolean`

Raise `NullArgument` – `repository_id` is `null`

compliance: mandatory – This method must be implemented.

`Repository.`**`get_assignable_repository_ids`**`(`*repository_id*`)`

Gets a list of repositories including and under the given repository node in which any composition can be assigned.

Parameters **`repository_id`** (`osid.id.Id`) – the `Id` of the `Repository`

Returns list of assignable repository `Ids`

Return type `osid.id.IdList`

Raise `NullArgument` – `repository_id` is `null`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`Repository.`**`get_assignable_repository_ids_for_composition`**(*repository_id*, *composition_id*)

Gets a list of repositories including and under the given repository node in which a specific composition can be assigned.

Parameters

- **`repository_id`** (`osid.id.Id`) – the `Id` of the `Repository`
- **`composition_id`** (`osid.id.Id`) – the `Id` of the `Composition`

Returns list of assignable repository `Ids`

Return type `osid.id.IdList`

Raise `NullArgument` – `repository_id` or `composition_id` is `null`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`Repository.`**`assign_composition_to_repository`**(*composition_id*, *repository_id*)

Adds an existing `Composition` to a `Repository`.

Parameters

- **`composition_id`** (`osid.id.Id`) – the `Id` of the `Composition`
- **`repository_id`** (`osid.id.Id`) – the `Id` of the `Repository`

Raise `AlreadyExists` – `composition_id` already assigned to `repository_id`

Raise `NotFound` – `composition_id` or `repository_id` not found

Raise `NullArgument` – `composition_id` or `repository_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Repository.`**`unassign_composition_from_repository`**(*composition_id*, *repository_id*)

Removes `Composition` from a `Repository`.

Parameters

- **`composition_id`** (`osid.id.Id`) – the `Id` of the `Composition`
- **`repository_id`** (`osid.id.Id`) – the `Id` of the `Repository`

Raise `NotFound` – `composition_id` or `repository_id` not found or `composition_id` not assigned to `repository_id`

Raise `NullArgument` – `composition_id` or `repository_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Objects

Asset

class `dlkit.repository.objects.`**`Asset`**

Bases: *`dlkit.osid.objects.OsidObject`*, *`dlkit.osid.markers.Aggregateable`*, *`dlkit.osid.markers.Sourceable`*

An `Asset` represents some digital content.

Example assets might be a text document, an image, or a movie. The content data, and metadata related directly to the content format and quality, is accessed through `AssetContent. Assets` , like all `OsidObjects,` include a type a record to qualify the `Asset` and include additional data. The division between the `Asset Type` and `AssetContent` is to separate data describing the asset from data describing the format of the contents, allowing a consumer to select among multiple formats, sizes or levels of fidelity.

An example is a photograph of the Bay Bridge. The content may deliver a JPEG in multiple resolutions where the `AssetContent` may also desribe size or compression factor for each one. The content may also include an uncompressed TIFF version. The `Asset Type` may be "photograph" indicating that the photo itself is the asset managed in this repository.

Since an Asset may have multiple `AssetContent` structures, the decision of how many things to stuff inside a single asset comes down to if the content is actually a different format, or size, or quality, falling under the same creator, copyright, publisher and distribution rights as the original. This may, in some cases, provide a means to implement some accessibility, it doesn't handle the case where, to meet an accessibility requirement, one asset needs to be substituted for another. The Repository OSID manages this aspect outside the scope of the core `Asset` definition.

`Assets` map to `AssetSubjects`. `AssetSubjects` are `OsidObjects` that capture a subject matter. In the above example, an `AssetSubject` may be defined for the Bay Bridge and include data describing the bridge. The single subject can map to multiple assets depicting the bridge providing a single entry for a search and a single place to describe a bridge. Bridges, as physical items, may also be described using the Resource OSID in which case the use of the `AssetSubject` acts as a cover for the underlying `Resource` to assist repository-only consumers.

The `Asset` definition includes some basic copyright and related licensing information to assist in finding free-to-use content, or to convey the distribution restrictions that may be placed on the asset. Generally, if no data is available it is to be assumed that all rights are reserved.

A publisher is applicable if the content of this `Asset` has been published. Not all `Assets` in this `Repository` may have a published status and such a status may effect the applicability of copyright law. To trace the source of an `Asset,` both a provider and source are defined. The provider indicates where this repository acquired the asset and the source indicates the original provider or copyright owner. In the case of a published asset, the source is the publisher.

`Assets` also define methods to facilitate searches over time and space as it relates to the subject matter. This may at times be redundant with the `AssetSubject`. In the case of the Bay Bridge photograph, the temporal coverage may include 1936, when it opened, and/or indicate when the photo was taken to capture a current event of the bridge. The decision largeley depends on what desired effect is from a search. The spatial coverage may describe the gps coordinates of the bridge or describe the spatial area encompassed in the view. In either case, a "photograph" type may unambiguously defined methods to describe the exact time the photograph was taken and the location of the photographer.

The core Asset defines methods to perform general searches and construct bibliographic entries without knowledge of a particular `Asset` or `AssetContent` record `Type`.

title

Gets the proper title of this asset.

This may be the same as the display name or the display name may be used for a less formal label.

Returns the title of this asset

Return type `osid.locale.DisplayText`

compliance: mandatory – This method must be implemented.

is_copyright_status_known()

Tests if the copyright status is known.

return `true` if the copyright status of this asset is known, `false` otherwise. If `false`, `is_public_domain()`, ‘‘can_distribute_verbatim(), can_distribute_alterations() and

can_distribute_compositions()‘‘ may also be `false`.

rtype `boolean`

compliance: mandatory – This method must be implemented.

is_public_domain()

Tests if this asset is in the public domain.

An asset is in the public domain if copyright is not applicable, the copyright has expired, or the copyright owner has expressly relinquished the copyright.

Returns `true` if this asset is in the public domain, `false` otherwise. If `true`, `can_distribute_verbatim()`, `can_distribute_alterations()` and `can_distribute_compositions()` must also be `true`.

Return type `boolean`

Raise `IllegalState` – `is_copyright_status_known()` is `false`

compliance: mandatory – This method must be implemented.

copyright_

Gets the copyright statement and of this asset which identifies the current copyright holder.

For an asset in the public domain, this method may return the original copyright statement although it may be no longer valid.

Returns the copyright statement or an empty string if none available. An empty string does not imply the asset is not protected by copyright.

Return type `osid.locale.DisplayText`

Raise `IllegalState` – `is_copyright_status_known()` is `false`

compliance: mandatory – This method must be implemented.

copyright_registration

Gets the copyright registration information for this asset.

Returns the copyright registration. An empty string means the registration status isn’t known.

Return type `string`

Raise `IllegalState` – `is_copyright_status_known()` is `false`

compliance: mandatory – This method must be implemented.

can_distribute_verbatim()

Tests if there are any license restrictions on this asset that restrict the distribution, re-publication or public display of this asset, commercial or otherwise, without modification, alteration, or inclusion in other works.

This method is intended to offer consumers a means of filtering out search results that restrict distribution for any purpose. The scope of this method does not include licensing that describes warranty disclaimers or attribution requirements. This method is intended for informational purposes only and does not replace or override the terms specified in a license agreement which may specify exceptions or additional restrictions.

Returns `true` if the asset can be distributed verbatim, `false` otherwise.

Return type `boolean`

Raise `IllegalState` – `is_copyright_status_known()` is `false`

compliance: mandatory – This method must be implemented.

can_distribute_alterations()

Tests if there are any license restrictions on this asset that restrict the distribution, re-publication or public display of any alterations or modifications to this asset, commercial or otherwise, for any purpose.

This method is intended to offer consumers a means of filtering out search results that restrict the distribution or public display of any modification or alteration of the content or its metadata of any kind, including editing, translation, resampling, resizing and cropping. The scope of this method does not include licensing that describes warranty disclaimers or attribution requirements. This method is intended for informational purposes only and does not replace or override the terms specified in a license agreement which may specify exceptions or additional restrictions.

Returns `true` if the asset can be modified, `false` otherwise. If `true`, `can_distribute_verbatim()` must also be `true`.

Return type `boolean`

Raise `IllegalState` – `is_copyright_status_known()` is `false`

compliance: mandatory – This method must be implemented.

can_distribute_compositions()

Tests if there are any license restrictions on this asset that restrict the distribution, re-publication or public display of this asset as an inclusion within other content or composition, commercial or otherwise, for any purpose, including restrictions upon the distribution or license of the resulting composition.

This method is intended to offer consumers a means of filtering out search results that restrict the use of this asset within compositions. The scope of this method does not include licensing that describes warranty disclaimers or attribution requirements. This method is intended for informational purposes only and does not replace or override the terms specified in a license agreement which may specify exceptions or additional restrictions.

Returns `true` if the asset can be part of a larger composition `false` otherwise. If `true`, `can_distribute_verbatim()` must also be `true`.

Return type `boolean`

Raise `IllegalState` – `is_copyright_status_known()` is `false`

compliance: mandatory – This method must be implemented.

source_id

Gets the `Resource Id` of the source of this asset.

The source is the original owner of the copyright of this asset and may differ from the creator of this asset. The source for a published book written by Margaret Mitchell would be Macmillan. The source for an unpublished painting by Arthur Goodwin would be Arthur Goodwin.

An `Asset` is `Sourceable` and also contains a provider identity. The provider is the entity that makes this digital asset available in this repository but may or may not be the publisher of the contents depicted in the asset. For example, a map published by Ticknor and Fields in 1848 may have a provider of Library of Congress and a source of Ticknor and Fields. If copied from a repository at Middlebury College, the provider would be Middlebury College and a source of Ticknor and Fields.

Returns the source `Id`

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`source`

Gets the `Resource` of the source of this asset.

The source is the original owner of the copyright of this asset and may differ from the creator of this asset. The source for a published book written by Margaret Mitchell would be Macmillan. The source for an unpublished painting by Arthur Goodwin would be Arthur Goodwin.

Returns the source

Return type `osid.resource.Resource`

compliance: mandatory – This method must be implemented.

`provider_link_ids`

Gets the resource `Ids` representing the source of this asset in order from the most recent provider to the originating source.

Returns the provider `Ids`

Return type `osid.id.IdList`

compliance: mandatory – This method must be implemented.

`provider_links`

Gets the `Resources` representing the source of this asset in order from the most recent provider to the originating source.

Returns the provider chain

Return type `osid.resource.ResourceList`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`created_date`

Gets the created date of this asset, which is generally not related to when the object representing the asset was created.

The date returned may indicate that not much is known.

Returns the created date

Return type `osid.calendaring.DateTime`

compliance: mandatory – This method must be implemented.

`is_published``()`

Tests if this asset has been published.

Not all assets viewable in this repository may have been published. The source of a published asset indicates the publisher.

Returns true if this asset has been published, `false` if unpublished or its published status is not known

Return type `boolean`

compliance: mandatory – This method must be implemented.

`published_date`

Gets the published date of this asset.

Unpublished assets have no published date. A published asset has a date available, however the date returned may indicate that not much is known.

Returns the published date

Return type `osid.calendaring.DateTime`

Raise `IllegalState` – `is_published()` is `false`

compliance: mandatory – This method must be implemented.

`principal_credit_string`

Gets the credits of the principal people involved in the production of this asset as a display string.

Returns the principal credits

Return type `osid.locale.DisplayText`

compliance: mandatory – This method must be implemented.

`asset_content_ids`

Gets the content `Ids` of this asset.

Returns the asset content `Ids`

Return type `osid.id.IdList`

compliance: mandatory – This method must be implemented.

`asset_contents`

Gets the content of this asset.

Returns the asset contents

Return type `osid.repository.AssetContentList`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`is_composition``()`

Tetss if this asset is a representation of a composition of assets.

Returns true if this asset is a composition, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`composition_id`

Gets the `Composition Id` corresponding to this asset.

Returns the composiiton `Id`

Return type `osid.id.Id`

Raise `IllegalState` – `is_composition()` is `false`

compliance: mandatory – This method must be implemented.

composition

Gets the Composition corresponding to this asset.

Returns the composiiton

Return type `osid.repository.Composition`

Raise `IllegalState` – `is_composition()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

get_asset_record(*asset_record_type*)

Gets the asset record corresponding to the given `Asset` record `Type`.

This method is used to retrieve an object implementing the requested record. The `asset_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(asset_record_type)` is `true` .

Parameters **asset_record_type** (`osid.type.Type`) – an asset record type

Returns the asset record

Return type `osid.repository.records.AssetRecord`

Raise `NullArgument` – `asset_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(asset_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Asset Form

class `dlkit.repository.objects.`**AssetForm**

Bases: *dlkit.osid.objects.OsidObjectForm*, *dlkit.osid.objects.OsidAggregateableForm*, *dlkit.osid.objects.OsidSourceableForm*

This is the form for creating and updating `Assets`.

Like all `OsidForm` objects, various data elements may be set here for use in the create and update methods in the `AssetAdminSession`. For each data element that may be set, metadata may be examined to provide display hints or data constraints.

title_metadata

Gets the metadata for an asset title.

Returns metadata for the title

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

title

public_domain_metadata

Gets the metadata for the public domain flag.

Returns metadata for the public domain

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`public_domain`

`copyright_metadata`

Gets the metadata for the copyright.

Returns metadata for the copyright

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`copyright_`

`copyright_registration_metadata`

Gets the metadata for the copyright registration.

Returns metadata for the copyright registration

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`copyright_registration`

`distribute_verbatim_metadata`

Gets the metadata for the distribute verbatim rights flag.

Returns metadata for the distribution rights fields

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`distribute_verbatim`

`distribute_alterations_metadata`

Gets the metadata for the distribute alterations rights flag.

Returns metadata for the distribution rights fields

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`distribute_alterations`

`distribute_compositions_metadata`

Gets the metadata for the distribute compositions rights flag.

Returns metadata for the distribution rights fields

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`distribute_compositions`

`source_metadata`

Gets the metadata for the source.

Returns metadata for the source

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`source`

provider_links_metadata
Gets the metadata for the provider chain.

Returns metadata for the provider chain

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

provider_links

created_date_metadata
Gets the metadata for the asset creation date.

Returns metadata for the created date

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

created_date

published_metadata
Gets the metadata for the published status.

Returns metadata for the published field

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

published

published_date_metadata
Gets the metadata for the published date.

Returns metadata for the published date

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

published_date

principal_credit_string_metadata
Gets the metadata for the principal credit string.

Returns metadata for the credit string

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

principal_credit_string

composition_metadata
Gets the metadata for linking this asset to a composition.

Returns metadata for the composition

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

composition

get_asset_form_record(*asset_record_type*)
Gets the `AssetFormRecord` corresponding to the given `Asset` record `Type`.

Parameters **`asset_record_type`** (`osid.type.Type`) – an asset record type

Returns the asset form record

Return type `osid.repository.records.AssetFormRecord`

Raise `NullArgument` – `asset_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(asset_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Asset List

class `dlkit.repository.objects.`**`AssetList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists`, `AssetList` provides a means for accessing `Asset` elements sequentially either one at a time or many at a time.

Examples: while (al.hasNext()) { Asset asset = al.getNextAsset(); }

or

while (al.hasNext()) { Asset[] assets = al.getNextAssets(al.available());

}

`next_asset`

Gets the next `Asset` in this list.

Returns the next `Asset` in this list. The `has_next()` method should be used to test that a next `Asset` is available before calling this method.

Return type `osid.repository.Asset`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_assets`(*n*)

Gets the next set of `Assets` in this list which must be less than or equal to the return from `available()`.

Parameters **`n`** (`cardinal`) – the number of `Asset` elements requested which must be less than or equal to `available()`

Returns an array of `Asset` elements.The length of the array is less than or equal to the number specified.

Return type `osid.repository.Asset`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Asset Content

class `dlkit.repository.objects.`**`AssetContent`**

Bases: *`dlkit.osid.objects.OsidObject`*, *`dlkit.osid.markers.Subjugateable`*

`AssetContent` represents a version of content represented by an `Asset`.

Although `AssetContent` is a separate `OsidObject` with its own `Id` to distuinguish it from other content inside an `Asset, AssetContent` can only be accessed through an `Asset`.

Once an `Asset` is selected, multiple contents should be negotiated using the size, fidelity, accessibility requirements or application evnironment.

`asset_id`

Gets the `Asset Id` corresponding to this content.

Returns the asset `Id`

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`asset`

Gets the `Asset` corresponding to this content.

Returns the asset

Return type `osid.repository.Asset`

compliance: mandatory – This method must be implemented.

`accessibility_types`

Gets the accessibility types associated with this content.

Returns list of content accessibility types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

`has_data_length``()`

Tests if a data length is available.

Returns `true` if a length is available for this content, `false` otherwise.

Return type `boolean`

compliance: mandatory – This method must be implemented.

`data_length`

Gets the length of the data represented by this content in bytes.

Returns the length of the data stream

Return type `cardinal`

Raise `IllegalState` – `has_data_length()` is `false`

compliance: mandatory – This method must be implemented.

`data`

Gets the asset content data.

Returns the length of the content data

Return type `osid.transport.DataInputStream`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

has_url()

Tests if a URL is associated with this content.

Returns `true` if a URL is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

url

Gets the URL associated with this content for web-based retrieval.

Returns the url for this data

Return type `string`

Raise `IllegalState` – `has_url()` is `false`

compliance: mandatory – This method must be implemented.

get_asset_content_record(*asset_content_content_record_type*)

Gets the asset content record corresponding to the given `AssetContent` record `Type`.

This method is used to retrieve an object implementing the requested record. The `asset_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(asset_record_type)` is `true` .

Parameters **asset_content_content_record_type** (`osid.type.Type`) – the type of the record to retrieve

Returns the asset content record

Return type `osid.repository.records.AssetContentRecord`

Raise `NullArgument` – `asset_content_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(asset_content_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Asset Content Form

class `dlkit.repository.objects.`**AssetContentForm**

Bases: *dlkit.osid.objects.OsidObjectForm*, *dlkit.osid.objects.OsidSubjugateableForm*

This is the form for creating and updating content for `AssetContent`.

Like all `OsidForm` objects, various data elements may be set here for use in the create and update methods in the `AssetAdminSession`. For each data element that may be set, metadata may be examined to provide display hints or data constraints.

accessibility_type_metadata

Gets the metadata for an accessibility type.

Returns metadata for the accessibility types

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

add_accessibility_type(*accessibility_type*)
Adds an accessibility type.

Multiple types can be added.

Parameters **accessibility_type** (`osid.type.Type`) – a new accessibility type

Raise `InvalidArgument` – `accessibility_type` is invalid

Raise `NoAccess` – `Metadata.isReadOnly()` is `true`

Raise `NullArgument` – `accessibility_t_ype` is `null`

compliance: mandatory – This method must be implemented.

remove_accessibility_type(*accessibility_type*)
Removes an accessibility type.

Parameters **accessibility_type** (`osid.type.Type`) – accessibility type to remove

Raise `NoAccess` – `Metadata.isReadOnly()` is `true`

Raise `NotFound` – acessibility type not found

Raise `NullArgument` – `accessibility_type` is `null`

compliance: mandatory – This method must be implemented.

accessibility_types

data_metadata
Gets the metadata for the content data.

Returns metadata for the content data

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

data

url_metadata
Gets the metadata for the url.

Returns metadata for the url

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

url

get_asset_content_form_record(*asset_content_record_type*)
Gets the `AssetContentFormRecord` corresponding to the given asset content record `Type`.

Parameters **asset_content_record_type** (`osid.type.Type`) – an asset content record type

Returns the asset content form record

Return type `osid.repository.records.AssetContentFormRecord`

Raise `NullArgument` – `asset_content_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(asset_content_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Asset Content List

class `dlkit.repository.objects.`**`AssetContentList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists`, `AssetContentList` provides a means for accessing `AssetContent` elements sequentially either one at a time or many at a time.

Examples: while (acl.hasNext()) { AssetContent content = acl.getNextAssetContent(); }

or

while (acl.hasNext()) { AssetContent[] contents = acl.getNextAssetContents(acl.available());

}

`next_asset_content`

Gets the next `AssetContent` in this list.

Returns the next `AssetContent` in this list. The `has_next()` method should be used to test that a next `AssetContent` is available before calling this method.

Return type `osid.repository.AssetContent`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_asset_contents`(*n*)

Gets the next set of `AssetContents` in this list which must be less than or equal to the return from `available()`.

Parameters **n** (`cardinal`) – the number of `AssetContent` elements requested which must be less than or equal to `available()`

Returns an array of `AssetContent` elements.The length of the array is less than or equal to the number specified.

Return type `osid.repository.AssetContent`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Composition

class `dlkit.repository.objects.`**`Composition`**

Bases: *`dlkit.osid.objects.OsidObject`*, *`dlkit.osid.markers.Containable`*, *`dlkit.osid.markers.Operable`*, *`dlkit.osid.markers.Sourceable`*

A `Composition` represents an authenticatable identity.

Like all OSID objects, a `Composition` is identified by its Id and any persisted references should use the Id.

`children_ids`

Gets the child `Ids` of this composition.

Returns the composition child `Ids`

Return type `osid.id.IdList`

compliance: mandatory – This method must be implemented.

`children`

Gets the children of this composition.

Returns the composition children

Return type `osid.repository.CompositionList`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_composition_record`(*composition_record_type*)

Gets the composition record corresponding to the given `Composition` record `Type`.

This method is used to retrieve an object implementing the requested record. The `composition_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(composition_record_type)` is `true` .

Parameters **`composition_record_type`** (`osid.type.Type`) – a composition record type

Returns the composition record

Return type `osid.repository.records.CompositionRecord`

Raise `NullArgument` – `composition_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(composition_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Composition Form

class `dlkit.repository.objects.`**`CompositionForm`**

Bases: *`dlkit.osid.objects.OsidObjectForm`*, *`dlkit.osid.objects.OsidContainableForm`*, *`dlkit.osid.objects.OsidOperableForm`*, *`dlkit.osid.objects.OsidSourceableForm`*

This is the form for creating and updating `Compositions`.

Like all `OsidForm` objects, various data elements may be set here for use in the create and update methods in the `CompositionAdminSession`. For each data element that may be set, metadata may be examined to provide display hints or data constraints.

`get_composition_form_record`(*composition_record_type*)

Gets the `CompositionFormRecord` corresponding to the given repository record `Type`.

Parameters **`composition_record_type`** (`osid.type.Type`) – a composition record type

Returns the composition form record

Return type `osid.repository.records.CompositionFormRecord`

Raise `NullArgument` – `composition_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(composition_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Composition List

class `dlkit.repository.objects.`**`CompositionList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists`, `CompositionList` provides a means for accessing `Composition` elements sequentially either one at a time or many at a time.

Examples: while (cl.hasNext()) { Composition composition = cl.getNextComposition(); }

or

while (cl.hasNext()) { Composition[] compositions = cl.getNextCompositions(cl.available());

}

`next_composition`

Gets the next `Composition` in this list.

Returns the next `Composition` in this list. The `has_next()` method should be used to test that a next `Composition` is available before calling this method.

Return type `osid.repository.Composition`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_compositions`(*n*)

Gets the next set of `Composition` elements in this list which must be less than or equal to the return from `available()`.

Parameters **n** (`cardinal`) – the number of `Composition` elements requested which must be less than or equal to `available()`

Returns an array of `Composition` elements.The length of the array is less than or equal to the number specified.

Return type `osid.repository.Composition`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Repository

class `dlkit.repository.objects.`**`Repository`**(*abc_repository_objects.Repository*, *osid_objects.OsidCatalog*)

`:noindex:`

`get_repository_record`(*repository_record_type*)

Gets the record corresponding to the given `Repository` record `Type`.

This method is used to retrieve an object implementing the requested record. The `repository_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(repository_record_type)` is `true` .

Parameters **`repository_record_type`** (`osid.type.Type`) – a repository record type

Returns the repository record

Return type `osid.repository.records.RepositoryRecord`

Raise `NullArgument` – `repository_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(repository_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Repository Form

class `dlkit.repository.objects.`**`RepositoryForm`**

Bases: *`dlkit.osid.objects.OsidCatalogForm`*

This is the form for creating and updating repositories.

Like all `OsidForm` objects, various data elements may be set here for use in the create and update methods in the `RepositoryAdminSession`. For each data element that may be set, metadata may be examined to provide display hints or data constraints.

`get_repository_form_record`(*repository_record_type*)

Gets the `RepositoryFormRecord` corresponding to the given repository record `Type`.

Parameters **`repository_record_type`** (`osid.type.Type`) – a repository record type

Returns the repository form record

Return type `osid.repository.records.RepositoryFormRecord`

Raise `NullArgument` – `repository_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(repository_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Repository List

class `dlkit.repository.objects.`**`RepositoryList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists`, `RepositoryList` provides a means for accessing `Repository` elements sequentially either one at a time or many at a time.

Examples: while (rl.hasNext()) { Repository repository = rl.getNextRepository(); }

or

while (rl.hasNext()) { Repository[] repositories = rl.getNextRepositories(rl.available());

}

next_repository

Gets the next `Repository` in this list.

Returns the next `Repository` in this list. The `has_next()` method should be used to test that a next `Repository` is available before calling this method.

Return type `osid.repository.Repository`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

get_next_repositories(*n*)

Gets the next set of `Repository` elements in this list which must be less than or equal to the return from `available()`.

Parameters **n** (`cardinal`) – the number of `Repository` elements requested which must be less than or equal to `available()`

Returns an array of `Repository` elements.The length of the array is less than or equal to the number specified.

Return type `osid.repository.Repository`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Repository Node

class `dlkit.repository.objects.`**RepositoryNode**

Bases: *`dlkit.osid.objects.OsidNode`*

This interface is a container for a partial hierarchy retrieval.

The number of hierarchy levels traversable through this interface depend on the number of levels requested in the `RepositoryHierarchySession`.

repository

Gets the `Repository` at this node.

Returns the repository represented by this node

Return type `osid.repository.Repository`

compliance: mandatory – This method must be implemented.

parent_repository_nodes

Gets the parents of this repository.

Returns the parents of the `id`

Return type `osid.repository.RepositoryNodeList`

compliance: mandatory – This method must be implemented.

child_repository_nodes

Gets the children of this repository.

Returns the children of this repository

Return type `osid.repository.RepositoryNodeList`

compliance: mandatory – This method must be implemented.

Repository Node List

class `dlkit.repository.objects.`**`RepositoryNodeList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists`, `RepositoryNodeList` provides a means for accessing `RepositoryNode` elements sequentially either one at a time or many at a time.

Examples: while (rnl.hasNext()) { RepositoryNode node = rnl.getNextRepositoryNode(); }

or

while (rnl.hasNext()) { RepositoryNode[] nodes = rnl.getNextRepositoryNodes(rnl.available());

}

`next_repository_node`

Gets the next `RepositoryNode` in this list.

Returns the next `RepositoryNode` in this list. The `has_next()` method should be used to test that a next `RepositoryNode` is available before calling this method.

Return type `osid.repository.RepositoryNode`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_repository_nodes`(*n*)

Gets the next set of `RepositoryNode` elements in this list which must be less than or equal to the return from `available()`.

Parameters n (`cardinal`) – the number of `RepositoryNode` elements requested which must be less than or equal to `available()`

Returns an array of `RepositoryNode` elements.The length of the array is less than or equal to the number specified.

Return type `osid.repository.RepositoryNode`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Queries

Asset Query

class `dlkit.repository.queries.`**`AssetQuery`**

Bases: *`dlkit.osid.queries.OsidObjectQuery`*, *`dlkit.osid.queries.OsidAggregateableQuery`*, *`dlkit.osid.queries.OsidSourceableQuery`*

This is the query for searching assets.

Each method specifies an `AND` term while multiple invocations of the same method produce a nested `OR`. The query record is identified by the `Asset Type`.

match_title (*title*, *string_match_type*, *match*)
Adds a title for this query.

Parameters

- **title** (`string`) – title string to match
- **string_match_type** (`osid.type.Type`) – the string match type
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `title` not of `string_match_type`

Raise `NullArgument` – `title` or `string_match_type` is `null`

Raise `Unsupported` – `supports_string_match_type(string_match_type)` is `false`

compliance: mandatory – This method must be implemented.

match_any_title (*match*)
Matches a title that has any value.

Parameters **match** (`boolean`) – `true` to match assets with any title, `false` to match assets with no title

compliance: mandatory – This method must be implemented.

title_terms

match_public_domain (*public_domain*)
Matches assets marked as public domain.

Parameters **public_domain** (`boolean`) – public domain flag

compliance: mandatory – This method must be implemented.

match_any_public_domain (*match*)
Matches assets with any public domain value.

Parameters **match** (`boolean`) – `true` to match assets with any public domain value, `false` to match assets with no public domain value

compliance: mandatory – This method must be implemented.

public_domain_terms

match_copyright (*copyright_*, *string_match_type*, *match*)
Adds a copyright for this query.

Parameters

- **copyright** (`string`) – copyright string to match
- **string_match_type** (`osid.type.Type`) – the string match type
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `copyright` not of `string_match_type`

Raise `NullArgument` – `copyright` or `string_match_type` is `null`

Raise `Unsupported` – `supports_string_match_type(string_match_type)` is `false`

compliance: mandatory – This method must be implemented.

match_any_copyright(*match*)
Matches assets with any copyright statement.

Parameters **match** (`boolean`) – `true` to match assets with any copyright value, `false` to match assets with no copyright value

compliance: mandatory – This method must be implemented.

copyright_terms

match_copyright_registration(*registration*, *string_match_type*, *match*)
Adds a copyright registration for this query.

Parameters

- **registration** (`string`) – copyright registration string to match
- **string_match_type** (`osid.type.Type`) – the string match type
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `registration` not of `string_match_type`

Raise `NullArgument` – `registration` or `string_match_type` is `null`

Raise `Unsupported` – `supports_string_match_type(string_match_type)` is `false`

compliance: mandatory – This method must be implemented.

match_any_copyright_registration(*match*)
Matches assets with any copyright registration.

Parameters **match** (`boolean`) – `true` to match assets with any copyright registration value, `false` to match assets with no copyright registration value

compliance: mandatory – This method must be implemented.

copyright_registration_terms

match_distribute_verbatim(*distributable*)
Matches assets marked as distributable.

Parameters **distributable** (`boolean`) – distribute verbatim rights flag

compliance: mandatory – This method must be implemented.

distribute_verbatim_terms

match_distribute_alterations(*alterable*)
Matches assets that whose alterations can be distributed.

Parameters **alterable** (`boolean`) – distribute alterations rights flag

compliance: mandatory – This method must be implemented.

distribute_alterations_terms

match_distribute_compositions(*composable*)
Matches assets that can be distributed as part of other compositions.

Parameters **composable** (`boolean`) – distribute compositions rights flag

compliance: mandatory – This method must be implemented.

distribute_compositions_terms

match_source_id(*source_id*, *match*)
Sets the source `Id` for this query.

Parameters

- **source_id** (`osid.id.Id`) – the source `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `source_id` is `null`

compliance: mandatory – This method must be implemented.

source_id_terms

supports_source_query()
Tests if a `ResourceQuery` is available for the source.

Returns `true` if a resource query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

source_query
Gets the query for the source.

Multiple queries can be retrieved for a nested `OR` term.

Returns the source query

Return type `osid.resource.ResourceQuery`

Raise `Unimplemented` – `supports_source_query()` is `false`

compliance: optional – This method must be implemented if ``supports_source_query()`` is ``true``.

match_any_source(*match*)
Matches assets with any source.

Parameters match (`boolean`) – `true` to match assets with any source, `false` to match assets with no sources

compliance: mandatory – This method must be implemented.

source_terms

match_created_date(*start*, *end*, *match*)
Match assets that are created between the specified time period.

Parameters

- **start** (`osid.calendaring.DateTime`) – start time of the query
- **end** (`osid.calendaring.DateTime`) – end time of the query
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `end` is les than `start`

Raise `NullArgument` – `start` or `end` is `null`

compliance: mandatory – This method must be implemented.

match_any_created_date(*match*)
Matches assets with any creation time.

Parameters match (`boolean`) – `true` to match assets with any created time, `false` to match assets with no cerated time

compliance: mandatory – This method must be implemented.

created_date_terms

match_published(*published*)

Marks assets that are marked as published.

Parameters published (boolean) – published flag

compliance: mandatory – This method must be implemented.

published_terms

match_published_date(*start*, *end*, *match*)

Match assets that are published between the specified time period.

Parameters

- **start** (osid.calendaring.DateTime) – start time of the query
- **end** (osid.calendaring.DateTime) – end time of the query
- **match** (boolean) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `end` is les than `start`

Raise `NullArgument` – `start` or `end` is `null`

compliance: mandatory – This method must be implemented.

match_any_published_date(*match*)

Matches assets with any published time.

Parameters match (boolean) – `true` to match assets with any published time, `false` to match assets with no published time

compliance: mandatory – This method must be implemented.

published_date_terms

match_principal_credit_string(*credit*, *string_match_type*, *match*)

Adds a principal credit string for this query.

Parameters

- **credit** (string) – credit string to match
- **string_match_type** (osid.type.Type) – the string match type
- **match** (boolean) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `credit` not of `string_match_type`

Raise `NullArgument` – `credit` or `string_match_type` is `null`

Raise `Unsupported` – `supports_string_match_type(string_match_type)` is `false`

compliance: mandatory – This method must be implemented.

match_any_principal_credit_string(*match*)

Matches a principal credit string that has any value.

Parameters match (boolean) – `true` to match assets with any principal credit string, `false` to match assets with no principal credit string

compliance: mandatory – This method must be implemented.

principal_credit_string_terms

match_temporal_coverage(*start*, *end*, *match*)
Match assets that whose coverage falls between the specified time period inclusive.

Parameters

- **start** (`osid.calendaring.DateTime`) – start time of the query
- **end** (`osid.calendaring.DateTime`) – end time of the query
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `end` is less than `start`

Raise `NullArgument` – `start` or `end` is `null`

compliance: mandatory – This method must be implemented.

match_any_temporal_coverage(*match*)
Matches assets with any temporal coverage.

Parameters **match** (`boolean`) – `true` to match assets with any temporal coverage, `false` to match assets with no temporal coverage

compliance: mandatory – This method must be implemented.

temporal_coverage_terms

match_location_id(*location_id*, *match*)
Sets the location `Id` for this query of spatial coverage.

Parameters

- **location_id** (`osid.id.Id`) – the location `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `location_id` is `null`

compliance: mandatory – This method must be implemented.

location_id_terms

supports_location_query()
Tests if a `LocationQuery` is available for the provider.

Returns `true` if a location query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

location_query
Gets the query for a location.

Multiple queries can be retrieved for a nested `OR` term.

Returns the location query

Return type `osid.mapping.LocationQuery`

Raise `Unimplemented` – `supports_location_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_location_query()‘‘ is ‘‘true‘‘.

match_any_location(*match*)
Matches assets with any provider.

Parameters **match** (`boolean`) – `true` to match assets with any location, `false` to match assets with no locations

compliance: mandatory – This method must be implemented.

location_terms

match_spatial_coverage(*spatial_unit*, *match*)
Matches assets that are contained within the given spatial unit.

Parameters

- **spatial_unit** (`osid.mapping.SpatialUnit`) – the spatial unit
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `spatial_unit` is `null`

Raise `Unsupported` – `spatial_unit` is not suppoted

compliance: mandatory – This method must be implemented.

spatial_coverage_terms

match_spatial_coverage_overlap(*spatial_unit*, *match*)
Matches assets that overlap or touch the given spatial unit.

Parameters

- **spatial_unit** (`osid.mapping.SpatialUnit`) – the spatial unit
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `spatial_unit` is `null`

Raise `Unsupported` – `spatial_unit` is not suppoted

compliance: mandatory – This method must be implemented.

match_any_spatial_coverage(*match*)
Matches assets with no spatial coverage.

Parameters **match** (`boolean`) – `true` to match assets with any spatial coverage, `false` to match assets with no spatial coverage

compliance: mandatory – This method must be implemented.

spatial_coverage_overlap_terms

match_asset_content_id(*asset_content_id*, *match*)
Sets the asset content `Id` for this query.

Parameters

- **asset_content_id** (`osid.id.Id`) – the asset content `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `asset_content_id` is `null`

compliance: mandatory – This method must be implemented.

asset_content_id_terms

supports_asset_content_query()
Tests if an `AssetContentQuery` is available.

Returns `true` if an asset content query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

asset_content_query
Gets the query for the asset content.

Multiple queries can be retrieved for a nested `OR` term.

Returns the asset contents query

Return type `osid.repository.AssetContentQuery`

Raise `Unimplemented` – `supports_asset_content_query()` is `false`

compliance: optional – This method must be implemented if ``supports_asset_content_query()`` is ``true``.

match_any_asset_content(*match*)
Matches assets with any content.

Parameters match (`boolean`) – `true` to match assets with any content, `false` to match assets with no content

compliance: mandatory – This method must be implemented.

asset_content_terms

match_composition_id(*composition_id*, *match*)
Sets the composition `Id` for this query to match assets that are a part of the composition.

Parameters

- **composition_id** (`osid.id.Id`) – the composition `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `composition_id` is `null`

compliance: mandatory – This method must be implemented.

composition_id_terms

supports_composition_query()
Tests if a `CompositionQuery` is available.

Returns `true` if a composition query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

composition_query
Gets the query for a composition.

Multiple queries can be retrieved for a nested `OR` term.

Returns the composition query

Return type `osid.repository.CompositionQuery`

Raise `Unimplemented` – `supports_composition_query()` is `false`

compliance: optional – This method must be implemented if ``supports_composition_query()`` is ``true``.

match_any_composition(*match*)
Matches assets with any composition mappings.

Parameters match (`boolean`) – `true` to match assets with any composition, `false` to match assets with no composition mappings

compliance: mandatory – This method must be implemented.

composition_terms

match_repository_id(*repository_id*, *match*)
Sets the repository `Id` for this query.

Parameters

- **repository_id** (`osid.id.Id`) – the repository `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `repository_id` is `null`

compliance: mandatory – This method must be implemented.

repository_id_terms

supports_repository_query()
Tests if a `RepositoryQuery` is available.

Returns `true` if a repository query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

repository_query
Gets the query for a repository.

Multiple queries can be retrieved for a nested `OR` term.

Returns the repository query

Return type `osid.repository.RepositoryQuery`

Raise `Unimplemented` – `supports_repository_query()` is `false`

compliance: optional – This method must be implemented if ``supports_repository_query()`` is ``true``.

repository_terms

get_asset_query_record(*asset_record_type*)
Gets the asset query record corresponding to the given `Asset` record `Type`.

Multiuple retrievals produce a nested `OR` term.

Parameters **asset_record_type** (`osid.type.Type`) – an asset record type

Returns the asset query record

Return type `osid.repository.records.AssetQueryRecord`

Raise `NullArgument` – `asset_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(asset_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Asset Content Query

class `dlkit.repository.queries.`**AssetContentQuery**
Bases: *dlkit.osid.queries.OsidObjectQuery*, *dlkit.osid.queries.OsidSubjugateableQuery*

This is the query for searching asset contents.

Each method forms an `AND` term while multiple invocations of the same method produce a nested `OR`.

match_accessibility_type (*accessibility_type*, *match*)
Sets the accessibility types for this query.

Supplying multiple types behaves like a boolean OR among the elements.

Parameters

- **accessibility_type** (`osid.type.Type`) – an accessibilityType
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `accessibility_type` is `null`

compliance: mandatory – This method must be implemented.

match_any_accessibility_type (*match*)
Matches asset content that has any accessibility type.

Parameters match (`boolean`) – `true` to match content with any accessibility type, `false` to match content with no accessibility type

compliance: mandatory – This method must be implemented.

accessibility_type_terms

match_data_length (*low*, *high*, *match*)
Matches content whose length of the data in bytes are inclusive of the given range.

Parameters

- **low** (`cardinal`) – low range
- **high** (`cardinal`) – high range
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `InvalidArgument` – `low` is greater than `high`

compliance: mandatory – This method must be implemented.

match_any_data_length (*match*)
Matches content that has any data length.

Parameters match (`boolean`) – `true` to match content with any data length, `false` to match content with no data length

compliance: mandatory – This method must be implemented.

data_length_terms

match_data (*data*, *match*, *partial*)
Matches data in this content.

Parameters

- **data** (`byte[]`) – list of matching strings
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match
- **partial** (`boolean`) – `true` for a partial match, `false` for a complete match

Raise `NullArgument` – `data` is `null`

compliance: mandatory – This method must be implemented.

match_any_data (*match*)
Matches content that has any data.

Parameters **match** (boolean) – true to match content with any data, false to match content with no data

compliance: mandatory – This method must be implemented.

data_terms

match_url (*url*, *string_match_type*, *match*)

Sets the url for this query.

Supplying multiple strings behaves like a boolean OR among the elements each which must correspond to the stringMatchType.

Parameters

- **url** (string) – url string to match
- **string_match_type** (osid.type.Type) – the string match type
- **match** (boolean) – true for a positive match, false for a negative match

Raise InvalidArgument – url not of string_match_type

Raise NullArgument – url or string_match_type is null

Raise Unsupported – supports_string_match_type(url) is false

compliance: mandatory – This method must be implemented.

match_any_url (*match*)

Matches content that has any url.

Parameters **match** (boolean) – true to match content with any url, false to match content with no url

compliance: mandatory – This method must be implemented.

url_terms

get_asset_content_query_record (*asset_content_record_type*)

Gets the asset content query record corresponding to the given AssetContent record Type.

Multiple record retrievals produce a nested OR term.

Parameters **asset_content_record_type** (osid.type.Type) – an asset content record type

Returns the asset content query record

Return type osid.repository.records.AssetContentQueryRecord

Raise NullArgument – asset_content_record_type is null

Raise OperationFailed – unable to complete request

Raise Unsupported – has_record_type(asset_content_record_type) is false

compliance: mandatory – This method must be implemented.

Composition Query

class dlkit.repository.queries.**CompositionQuery**

Bases: *dlkit.osid.queries.OsidObjectQuery*, *dlkit.osid.queries.OsidContainableQuery*, *dlkit.osid.queries.OsidOperableQuery*, *dlkit.osid.queries.OsidSourceableQuery*

This is the query for searching compositions.

Each method specifies an `AND` term while multiple invocations of the same method produces a nested `OR`.

`match_asset_id`(*asset_id*, *match*)
Sets the asset `Id` for this query.

Parameters

- **`asset_id`** (`osid.id.Id`) – the asset `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `asset_id` is `null`

compliance: mandatory – This method must be implemented.

`asset_id_terms`

`supports_asset_query`()
Tests if an `AssetQuery` is available.

Returns `true` if an asset query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`asset_query`
Gets the query for an asset.

Multiple retrievals produce a nested `OR` term.

Returns the asset query

Return type `osid.repository.AssetQuery`

Raise `Unimplemented` – `supports_asset_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_asset_query()‘‘ is ‘‘true‘‘.

`match_any_asset`(*match*)
Matches compositions that has any asset mapping.

Parameters **`match`** (`boolean`) – `true` to match compositions with any asset, `false` to match compositions with no asset

compliance: mandatory – This method must be implemented.

`asset_terms`

`match_containing_composition_id`(*composition_id*, *match*)
Sets the composition `Id` for this query to match compositions that have the specified composition as an ancestor.

Parameters

- **`composition_id`** (`osid.id.Id`) – a composition `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `composition_id` is `null`

compliance: mandatory – This method must be implemented.

`containing_composition_id_terms`

`supports_containing_composition_query`()
Tests if an `CompositionQuery` is available.

Returns `true` if a composition query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`containing_composition_query`

Gets the query for a composition.

Multiple retrievals produce a nested `OR` term.

Returns the composition query

Return type `osid.repository.CompositionQuery`

Raise `Unimplemented` – `supports_containing_composition_query()` is `false`

compliance: optional – This method must be implemented if ``supports_containing_composition_query()`` is ``true``.

`match_any_containing_composition`(*match*)

Matches compositions with any ancestor.

Parameters **`match`** (`boolean`) – `true` to match composition with any ancestor, `false` to match root compositions

compliance: mandatory – This method must be implemented.

`containing_composition_terms`

`match_contained_composition_id`(*composition_id*, *match*)

Sets the composition `Id` for this query to match compositions that contain the specified composition.

Parameters

- **`composition_id`** (`osid.id.Id`) – a composition `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `composition_id` is `null`

compliance: mandatory – This method must be implemented.

`contained_composition_id_terms`

`supports_contained_composition_query`()

Tests if an `CompositionQuery` is available.

Returns `true` if a composition query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`contained_composition_query`

Gets the query for a composition.

Multiple retrievals produce a nested `OR` term.

Returns the composition query

Return type `osid.repository.CompositionQuery`

Raise `Unimplemented` – `supports_contained_composition_query()` is `false`

compliance: optional – This method must be implemented if ``supports_contained_composition_query()`` is ``true``.

match_any_contained_composition(*match*)
Matches compositions that contain any other compositions.

Parameters **match** (boolean) – true to match composition with any descendant, false to match leaf compositions

compliance: mandatory – This method must be implemented.

contained_composition_terms

match_repository_id(*repository_id*, *match*)
Sets the repository Id for this query.

Parameters

- **repository_id** (osid.id.Id) – the repository Id
- **match** (boolean) – true for a positive match, false for a negative match

Raise NullArgument – repository_id is null

compliance: mandatory – This method must be implemented.

repository_id_terms

supports_repository_query()
Tests if a RepositoryQuery is available.

Returns true if a repository query is available, false otherwise

Return type boolean

compliance: mandatory – This method must be implemented.

repository_query
Gets the query for a repository.

Multiple queries can be retrieved for a nested OR term.

Returns the repository query

Return type osid.repository.RepositoryQuery

Raise Unimplemented – supports_repository_query() is false

compliance: optional – This method must be implemented if ``supports_repository_query()`` is ``true``.

repository_terms

get_composition_query_record(*composition_record_type*)
Gets the composition query record corresponding to the given Composition record Type.

Multiple retrievals produce a nested OR term.

Parameters **composition_record_type** (osid.type.Type) – a composition record type

Returns the composition query record

Return type osid.repository.records.CompositionQueryRecord

Raise NullArgument – composition_record_type is null

Raise OperationFailed – unable to complete request

Raise Unsupported – has_record_type(composition_record_type) is false

compliance: mandatory – This method must be implemented.

Repository Query

class `dlkit.repository.queries.`**`RepositoryQuery`**

Bases: *`dlkit.osid.queries.OsidCatalogQuery`*

This is the query for searching repositories.

Each method specifies an `AND` term while multiple invocations of the same method produce a nested `OR`.

`match_asset_id`(*asset_id*, *match*)

Sets the asset `Id` for this query.

Parameters

- **`asset_id`** (`osid.id.Id`) – an asset `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `asset_id` is `null`

compliance: mandatory – This method must be implemented.

`asset_id_terms`

`supports_asset_query`()

Tests if an `AssetQuery` is available.

Returns `true` if an asset query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`asset_query`

Gets the query for an asset.

Multiple retrievals produce a nested `OR` term.

Returns the asset query

Return type `osid.repository.AssetQuery`

Raise `Unimplemented` – `supports_asset_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_asset_query()‘‘ is ‘‘true‘‘.

`match_any_asset`(*match*)

Matches repositories that has any asset mapping.

Parameters **`match`** (`boolean`) – `true` to match repositories with any asset, `false` to match repositories with no asset

compliance: mandatory – This method must be implemented.

`asset_terms`

`match_composition_id`(*composition_id*, *match*)

Sets the composition `Id` for this query.

Parameters

- **`composition_id`** (`osid.id.Id`) – a composition `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `composition_id` is `null`

compliance: mandatory – This method must be implemented.

composition_id_terms

supports_composition_query()

Tests if a `CompositionQuery` is available.

Returns `true` if a composition query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

composition_query

Gets the query for a composition.

Multiple retrievals produce a nested `OR` term.

Returns the composition query

Return type `osid.repository.CompositionQuery`

Raise `Unimplemented` – `supports_composition_query()` is `false`

compliance: optional – This method must be implemented if ``supports_composition_query()`` is ``true``.

match_any_composition(*match*)

Matches repositories that has any composition mapping.

Parameters **match** (`boolean`) – `true` to match repositories with any composition, `false` to match repositories with no composition

compliance: mandatory – This method must be implemented.

composition_terms

match_ancestor_repository_id(*repository_id*, *match*)

Sets the repository `Id` for this query to match repositories that have the specified repository as an ancestor.

Parameters

- **repository_id** (`osid.id.Id`) – a repository `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `repository_id` is `null`

compliance: mandatory – This method must be implemented.

ancestor_repository_id_terms

supports_ancestor_repository_query()

Tests if a `RepositoryQuery` is available.

Returns `true` if a repository query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

ancestor_repository_query

Gets the query for a repository.

Multiple retrievals produce a nested `OR` term.

Returns the repository query

Return type `osid.repository.RepositoryQuery`

Raise `Unimplemented` – `supports_ancestor_repository_query()` is `false` *compliance: optional – This method must be implemented if ``supports_ancestor_repository_query()`` is ``true``.*

`match_any_ancestor_repository`(*match*)

Matches repositories with any ancestor.

Parameters **`match`** (`boolean`) – `true` to match repositories with any ancestor, `false` to match root repositories

compliance: mandatory – This method must be implemented.

`ancestor_repository_terms`

`match_descendant_repository_id`(*repository_id*, *match*)

Sets the repository `Id` for this query to match repositories that have the specified repository as a descendant.

Parameters

- **`repository_id`** (`osid.id.Id`) – a repository `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `repository_id` is `null`

compliance: mandatory – This method must be implemented.

`descendant_repository_id_terms`

`supports_descendant_repository_query`()

Tests if a `RepositoryQuery` is available.

Returns `true` if a repository query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`descendant_repository_query`

Gets the query for a repository.

Multiple retrievals produce a nested `OR` term.

Returns the repository query

Return type `osid.repository.RepositoryQuery`

Raise `Unimplemented` – `supports_descendant_repository_query()` is `false`

compliance: optional – This method must be implemented if ``supports_descendant_repository_query()`` is ``true``.

`match_any_descendant_repository`(*match*)

Matches repositories with any descendant.

Parameters **`match`** (`boolean`) – `true` to match repositories with any descendant, `false` to match leaf repositories

compliance: mandatory – This method must be implemented.

`descendant_repository_terms`

`get_repository_query_record`(*repository_record_type*)

Gets the repository query record corresponding to the given `Repository` record `Type`.

Multiple record retrievals produce a nested `OR` term.

Parameters **`repository_record_type`** (`osid.type.Type`) – a repository record type

Returns the repository query record

Return type `osid.repository.records.RepositoryQueryRecord`

Raise `NullArgument` – `repository_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(repository_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Records

Asset Record

class `dlkit.repository.records.`**`AssetRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `Asset`.

The methods specified by the record type are available through the underlying object.

Asset Query Record

class `dlkit.repository.records.`**`AssetQueryRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `AssetQuery`.

The methods specified by the record type are available through the underlying object.

Asset Form Record

class `dlkit.repository.records.`**`AssetFormRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `AssetForm`.

The methods specified by the record type are available through the underlying object.

Asset Search Record

class `dlkit.repository.records.`**`AssetSearchRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `AssetSearch`.

The methods specified by the record type are available through the underlying object.

Asset Content Record

class `dlkit.repository.records.`**`AssetContentRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `AssetContent`.

The methods specified by the record type are available through the underlying object.

Asset Content Query Record

class `dlkit.repository.records.`**`AssetContentQueryRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `AssetContentQuery`.

The methods specified by the record type are available through the underlying object.

Asset Content Form Record

class `dlkit.repository.records.`**`AssetContentFormRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for an `AssetForm`.

The methods specified by the record type are available through the underlying object.

Composition Record

class `dlkit.repository.records.`**`CompositionRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `Composition`.

The methods specified by the record type are available through the underlying object.

Composition Query Record

class `dlkit.repository.records.`**`CompositionQueryRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `CompositionQuery`.

The methods specified by the record type are available through the underlying object.

Composition Form Record

class `dlkit.repository.records.`**`CompositionFormRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `CompositionForm`.

The methods specified by the record type are available through the underlying object.

Composition Search Record

class `dlkit.repository.records.`**`CompositionSearchRecord`**
Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `CompositionSearch`.

The methods specified by the record type are available through the underlying object.

Repository Record

class `dlkit.repository.records.`**`RepositoryRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `Repository`.

The methods specified by the record type are available through the underlying object.

Repository Query Record

class `dlkit.repository.records.`**`RepositoryQueryRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `RepositoryQuery`.

The methods specified by the record type are available through the underlying object.

Repository Form Record

class `dlkit.repository.records.`**`RepositoryFormRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `RepositoryForm`.

The methods specified by the record type are available through the underlying object.

Repository Search Record

class `dlkit.repository.records.`**`RepositorySearchRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `RepositorySearch`.

The methods specified by the record type are available through the underlying object.

Resource

Summary

Resource Open Service Interface Definitions resource version 3.0.0

The Resource OSID defines a service to access and manage a directory of objects.

Resources

`Resources` may represent people, places or a set or arbitrary entities that are used throughout the OSIDs as references to indirect objects. In core OSID, `Resources` have no other meaning other than to provide an identifier and a relation to an authentication principal. `Resource Types` may define extra data to define an employee, organizational unit or an inventory item.

`Resources` are referenced throughout the OSIDs to and the abstraction level of this service provides a consistent interface with which to access and manage object references not directly pertinent to the service in play. For example, a Repository OSID may reference `Resources` as authors or a Course OSID may reference `Resources` for students and instructors. Each of these OSIDs may orchestrate a Resource OSID to provide management of the set of referenced resources.

A `Resource` genus Type may be used to provide a label the kind of resource. This service offers the flexibility that the producer of a film may be a person, a production company, or a fire hydrant. While genus `Types` may be used to constrain the kinds of `Resources` that may be related to various `OsidObjects` if necessary `,` OSID Consumers are expected to simply use the Resource as a reference. If an OSID Consumer wishes to provide a mechanism for updating a `Resource` referenced, the OSID Consumer should use an orchestrated Resource OSID.

Agents

A `Resource` also provides the mapping between an authentication `Agent` and the entity on whose behalf the agent is acting. An `Agent` can only map to a single `Resource` while a `Resource` can have multiple `Agents`. An agent that represents the unix login of "vijay" on server due.mit.edu can map to a `Resource` representing Vijay Kumar, who may also have a campus agent of "vkumar@mit.edu."

Group

When a `Resource` is referenced in another OSID, it is a singular entity. To provide groupings of multiple people or things, a `Resource` can also be defined as a hierarchical group of other resources. Whether a resource is a single entity or a group is an attribute of the `Resource` itself. If a `Resource` is a group, then its membership can be queried or managed in one of the group sessions. This overloading of the object definition serves to keep the nature of the resource separate from the other OSIDs such that a message to a "group", for example, is referenced as a single resource receipient. Other OSIDs are blind to whether or not a referenced `Resource` is a group or a singular entity..

Resource Relationships

For kicks, `Resources` may have arbitrrary relationships to other `Resources` using the `ResourceRelationship` interface. Resource relationships may also be used to provide a place to describe in more detail, or hang data, on a member to group relationship.

Bin Cataloging

`Resources` may be mapped into hierarchial `Bins` for the purpose of cataloging or federation.

Sub Packages

The Resource OSID includes a Resource Demographic OSID for managing dynamically generated populations of `Resources` and a Resource Batch OSID for managing `Resources` in bulk. Resource Open Service Interface Definitions resource version 3.0.0

The Resource OSID defines a service to access and manage a directory of objects.

Resources

`Resources` may represent people, places or a set or arbitrary entities that are used throughout the OSIDs as references to indirect objects. In core OSID, `Resources` have no other meaning other than to provide an identifier and a relation to an authentication principal. `Resource Types` may define extra data to define an employee, organizational unit or an inventory item.

`Resources` are referenced throughout the OSIDs to and the abstraction level of this service provides a consistent interface with which to access and manage object references not directly pertinent to the service in play. For example, a Repository OSID may reference `Resources` as authors or a Course OSID may reference `Resources` for students and instructors. Each of these OSIDs may orchestrate a Resource OSID to provide management of the set of referenced resources.

A `Resource` genus Type may be used to provide a label the kind of resource. This service offers the flexibility that the producer of a film may be a person, a production company, or a fire hydrant. While genus `Types` may be used to constrain the kinds of `Resources` that may be related to various `OsidObjects` if necessary `,` OSID Consumers are expected to simply use the Resource as a reference. If an OSID Consumer wishes to provide a mechanism for updating a `Resource` referenced, the OSID Consumer should use an orchestrated Resource OSID.

Agents

A `Resource` also provides the mapping between an authentication `Agent` and the entity on whose behalf the agent is acting. An `Agent` can only map to a single `Resource` while a `Resource` can have multiple `Agents`. An agent that represents the unix login of "vijay" on server due.mit.edu can map to a `Resource` representing Vijay Kumar, who may also have a campus agent of "vkumar@mit.edu."

Group

When a `Resource` is referenced in another OSID, it is a singular entity. To provide groupings of multiple people or things, a `Resource` can also be defined as a hierarchical group of other resources. Whether a resource is a single entity or a group is an attribute of the `Resource` itself. If a `Resource` is a group, then its membership can be queried or managed in one of the group sessions. This overloading of the object definition serves to keep the nature of the resource separate from the other OSIDs such that a message to a "group", for example, is referenced as a single resource receipient. Other OSIDs are blind to whether or not a referenced `Resource` is a group or a singular entity..

Resource Relationships

For kicks, `Resources` may have arbitrrary relationships to other `Resources` using the `ResourceRelationship` interface. Resource relationships may also be used to provide a place to describe in more detail, or hang data, on a member to group relationship.

Bin Cataloging

`Resources` may be mapped into hierarchial `Bins` for the purpose of cataloging or federation.

Sub Packages

The Resource OSID includes a Resource Demographic OSID for managing dynamically generated populations of `Resources` and a Resource Batch OSID for managing `Resources` in bulk.

Service Managers

Resource Profile

class `dlkit.services.resource.`**`ResourceProfile`**

Bases: `dlkit.osid.managers.OsidProfile`

The resource profile describes interoperability among resource services.

`supports_resource_lookup``()`

Tests if resource lookup is supported.

Returns `true` if resource lookup is supported , `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_resource_query``()`

Tests if resource query is supported.

Returns `true` if resource query is supported , `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_resource_search``()`

Tests if resource search is supported.

Returns `true` if resource search is supported , `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_resource_admin()
Tests if resource administration is supported.

Returns `true` if resource administration is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_resource_notification()
Tests if resource notification is supported.

Messages may be sent when resources are created, modified, or deleted.

Returns `true` if resource notification is supported `,` `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_resource_bin()
Tests if retrieving mappings of resource and bins is supported.

Returns `true` if resource bin mapping retrieval is supported `,` `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_resource_bin_assignment()
Tests if managing mappings of resource and bins is supported.

Returns `true` if resource bin assignment is supported `,` `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_resource_agent()
Tests if retrieving mappings of resource and agents is supported.

Returns `true` if resource agent mapping retrieval is supported `,` `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_resource_agent_assignment()
Tests if managing mappings of resources and agents is supported.

Returns `true` if resource agent assignment is supported `,` `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_bin_lookup()
Tests if bin lookup is supported.

Returns `true` if bin lookup is supported `,` `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

supports_bin_query()
Tests if bin query is supported.

Returns `true` if bin query is supported , `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_bin_admin`()

Tests if bin administration is supported.

Returns `true` if bin administration is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_bin_hierarchy`()

Tests if a bin hierarchy traversal is supported.

Returns `true` if a bin hierarchy traversal is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`supports_bin_hierarchy_design`()

Tests if a bin hierarchy design is supported.

Returns `true` if a bin hierarchy design is supported, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`resource_record_types`

Gets all the resource record types supported.

Returns the list of supported resource record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

`resource_search_record_types`

Gets all the resource search record types supported.

Returns the list of supported resource search record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

`resource_relationship_record_types`

Gets the supported `ResourceRelationship` record types.

Returns a list containing the supported `ResourceRelationship` record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

`resource_relationship_search_record_types`

Gets the supported `ResourceRelationship` search record types.

Returns a list containing the supported `ResourceRelationship` search record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

bin_record_types

Gets all the bin record types supported.

Returns the list of supported bin record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

bin_search_record_types

Gets all the bin search record types supported.

Returns the list of supported bin search record types

Return type `osid.type.TypeList`

compliance: mandatory – This method must be implemented.

Resource Manager

class `dlkit.services.resource.`**`ResourceManager`**(*proxy=None*)

Bases: `dlkit.osid.managers.OsidManager`, `dlkit.osid.sessions.OsidSession`, *`dlkit.services.resource.ResourceProfile`*

The resource manager provides access to resource lookup and creation sessions and provides interoperability tests for various aspects of this service.

The sessions included in this manager are:

- `ResourceLookupSession`: a session to retrieve resources
- `ResourceQuerySession`: a session to query resources
- `ResourceSearchSession`: a session to search for resources
- `ResourceAdminSession`: a session to create and delete resources
- `ResourceNotificationSession`: a session to receive notifications pertaining to resource changes
- `ResourceBinSession`: a session to look up resource to bin mappings
- `ResourceBinAssignmentSession`: a session to manage resource to bin mappings
- `ResourceSmartBinSession`: a session to manage smart resource bins
- `MembershipSession`: a session to query memberships
- `GroupSession`: a session to retrieve group memberships
- `GroupAssignmentSession`: a session to manage groups
- `GroupNotificationSession`: a session to retrieve notifications on changes to group membership
- `GroupHierarchySession`: a session to view a group hierarchy
- `RsourceAgentSession`: a session to retrieve `Resource` and `Agent` mappings
- `ResourceAgentAssignmentSession`: a session to manage `Resource` and `Agent` mappings
- `ResourceRelationshipLookupSession`: a session to retrieve resource relationships
- `ResourceRelationshipQuerySession`: a session to query for resource relationships
- `ResourceRelationshipSearchSession`: a session to search for resource relationships
- `ResourceRelationshipAdminSession`: a session to create and delete resource relationships

- `ResourceRelationshipNotificationSession`: a session to receive notifications pertaining to resource relationshipchanges
- `ResourceRelationshipBinSession`: a session to look up resource relationship to bin mappings
- `ResourceRelationshipBinAssignmentSession`: a session to manage resource relationship to bin mappings
- `ResourceRelationshipSmartBinSession`: a session to manage smart resource relationship bins
- `BinLookupSession`: a session to retrieve bins
- `BinQuerySession`: a session to query bins
- `BinSearchSession`: a session to search for bins
- `BinAdminSession`: a session to create, update and delete bins
- `BinNotificationSession`: a session to receive notifications pertaining to changes in bins
- `BinHierarchySession`: a session to traverse bin hierarchies
- `BinHierarchyDesignSession`: a session to manage bin hierarchies

resource_batch_manager

Gets the `ResourceBatchManager`.

Returns a `ResourceBatchManager`

Return type `osid.resource.batch.ResourceBatchManager`

Raise `OperationFailed` – unable to complete request

Raise `Unimplemented` – `supports_resource_batch()` is `false`

compliance: optional – This method must be implemented if ``supports_resource_batch()`` is ``true``.

resource_demographic_manager

Gets the `ResourceDemographicManager`.

Returns a `ResourceDemographicManager`

Return type `osid.resource.demographic.ResourceDemographicManager`

Raise `OperationFailed` – unable to complete request

Raise `Unimplemented` – `supports_resource_demographic()` is `false`

compliance: optional – This method must be implemented if ``supports_resource_demographic()`` is ``true``.

Bin Lookup Methods

`ResourceManager.`**can_lookup_bins**`()`

Tests if this user can perform `Bin` lookups.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if lookup methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`use_comparative_bin_view`**`()`

The returns from the bin methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`ResourceManager.`**`use_plenary_bin_view`**`()`

A complete view of the `Bin` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`ResourceManager.`**`get_bin`**`(`*bin_id*`)`

Gets the `Bin` specified by its `Id`.

In plenary mode, the exact `Id` is found or a `NotFound` results. Otherwise, the returned `Bin` may have a different `Id` than requested, such as the case where a duplicate `Id` was assigned to a `Bin` and retained for compatibility.

Parameters **`bin_id`** (`osid.id.Id`) – `Id` of the `Bin`

Returns the bin

Return type `osid.resource.Bin`

Raise `NotFound` – `bin_id` not found

Raise `NullArgument` – `bin_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method is must be implemented.

`ResourceManager.`**`get_bins_by_ids`**`(`*bin_ids*`)`

Gets a `BinList` corresponding to the given `IdList`.

In plenary mode, the returned list contains all of the bins specified in the `Id` list, in the order of the list, including duplicates, or an error results if an `Id` in the supplied list is not found or inaccessible. Otherwise, inaccessible `Bins` may be omitted from the list and may present the elements in any order including returning a unique set.

Parameters **`bin_ids`** (`osid.id.IdList`) – the list of `Ids` to retrieve

Returns the returned `Bin list`

Return type `osid.resource.BinList`

Raise `NotFound` – an `Id was` not found

Raise `NullArgument` – `bin_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`get_bins_by_genus_type`**`(`*bin_genus_type*`)`

Gets a `BinList` corresponding to the given bin genus `Type` which does not include bins of types derived from the specified `Type`.

In plenary mode, the returned list contains all known bins or an error results. Otherwise, the returned list may contain only those bins that are accessible through this session.

Parameters **`bin_genus_type`** (`osid.type.Type`) – a bin genus type

Returns the returned `Bin list`

Return type `osid.resource.BinList`

Raise `NullArgument` – `bin_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`get_bins_by_parent_genus_type`**(*bin_genus_type*)

Gets a `BinList` corresponding to the given bin genus `Type` and include any additional bins with genus types derived from the specified `Type`.

In plenary mode, the returned list contains all known bins or an error results. Otherwise, the returned list may contain only those bins that are accessible through this session.

Parameters **`bin_genus_type`** (`osid.type.Type`) – a bin genus type

Returns the returned `Bin list`

Return type `osid.resource.BinList`

Raise `NullArgument` – `bin_genus_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`get_bins_by_record_type`**(*bin_record_type*)

Gets a `BinList` containing the given bin record `Type`.

In plenary mode, the returned list contains all known bins or an error results. Otherwise, the returned list may contain only those bins that are accessible through this session.

Parameters **`bin_record_type`** (`osid.type.Type`) – a bin record type

Returns the returned `Bin list`

Return type `osid.resource.BinList`

Raise `NullArgument` – `bin_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`get_bins_by_provider`**(*resource_id*)

Gets a `BinList` from the given provider.

In plenary mode, the returned list contains all known bins or an error results. Otherwise, the returned list may contain only those bins that are accessible through this session.

Parameters **`resource_id`** (`osid.id.Id`) – a resource `Id`

Returns the returned `Bin list`

Return type `osid.resource.BinList`

Raise `NullArgument` – `resource_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`bins`**

Gets all `Bins`.

In plenary mode, the returned list contains all known bins or an error results. Otherwise, the returned list may contain only those bins that are accessible through this session.

Returns a list of `Bins`

Return type `osid.resource.BinList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Bin Query Methods

`ResourceManager.`**`can_search_bins`**`()`

Tests if this user can perform `Bin` searches.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer search operations to unauthorized users.

Returns `false` if search methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`bin_query`**

Gets a bin query.

The returned query will not have an extension query.

Returns the bin query

Return type `osid.resource.BinQuery`

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`get_bins_by_query`**`(`*bin_query*`)`

Gets a list of `Bins` matching the given bin query.

Parameters **`bin_query`** (`osid.resource.BinQuery`) – the bin query

Returns the returned `BinList`

Return type `osid.resource.BinList`

Raise `NullArgument` – `bin_query` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – a `bin_query` is not of this service

compliance: mandatory – This method must be implemented.

Bin Admin Methods

`ResourceManager.`**`can_create_bins`**`()`

Tests if this user can create `Bins`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known creating a `Bin` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer create operations to unauthorized users.

Returns `false` if `Bin` creation is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`can_create_bin_with_record_types`**`(`*bin_record_types*`)`

Tests if this user can create a single `Bin` using the desired record types.

While `ResourceManager.getBinRecordTypes()` can be used to examine which records are supported, this method tests which record(s) are required for creating a specific `Bin`. Providing an empty array tests if a `Bin` can be created with no records.

Parameters **`bin_record_types`** (`osid.type.Type[]`) – array of bin record types

Returns `true` if `Bin` creation using the specified `Types` is supported, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `bin_record_types` is `null`

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`get_bin_form_for_create`**`(`*bin_record_types*`)`

Gets the bin form for creating new bins.

Parameters **`bin_record_types`** (`osid.type.Type[]`) – array of bin record types

Returns the bin form

Return type `osid.resource.BinForm`

Raise `NullArgument` – `bin_record_types` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – unable to get form with requested record types

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`create_bin`**`(`*bin_form*`)`

Creates a new `Bin`.

Parameters **`bin_form`** (`osid.resource.BinForm`) – the form for this `Bin`

Returns the new `Bin`

Return type `osid.resource.Bin`

Raise `IllegalState` – `bin_form` already used in a create transaction

Raise `InvalidArgument` – one or more of the form elements is invalid

Raise `NullArgument` – `bin_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `bin_form` did not originate from `get_bin_form_for_create()`

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`can_update_bins`**`()`

Tests if this user can update `Bins`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating a `Bin` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer update operations to unauthorized users.

Returns `false` if `Bin` modification is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`get_bin_form_for_update`**`(`*bin_id*`)`

Gets the bin form for updating an existing bin.

A new bin form should be requested for each update transaction.

Parameters **`bin_id`** (`osid.id.Id`) – the `Id` of the `Bin`

Returns the bin form

Return type `osid.resource.BinForm`

Raise `NotFound` – `bin_id` is not found

Raise `NullArgument` – `bin_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`update_bin`**`(`*bin_form*`)`

Updates an existing bin.

Parameters **`bin_form`** (`osid.resource.BinForm`) – the form containing the elements to be updated

Raise `IllegalState` – `bin_form` already used in an update transaction

Raise `InvalidArgument` – the form contains an invalid value

Raise `NullArgument` – `bin_id` or `bin_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `bin_form` did not originate from `get_bin_form_for_update()`

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`can_delete_bins`**`()`

Tests if this user can delete `Bins`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting a `Bin` will result in a `PermissionDenied`. This is intended as a hint to an application that may not wish to offer delete operations to unauthorized users.

Returns `false` if `Bin` deletion is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`delete_bin`**`(`*bin_id*`)`

Deletes a `Bin`.

Parameters **`bin_id`** (`osid.id.Id`) – the `Id` of the `Bin` to remove

Raise `NotFound` – `bin_id` not found

Raise `NullArgument` – `bin_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`can_manage_bin_aliases`**`()`

Tests if this user can manage `Id` aliases for `Bins`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known changing an alias will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer alias operations to an unauthorized user.

Returns `false` if `Bin` aliasing is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`alias_bin`**`(`*bin_id*, *alias_id*`)`

Adds an `Id` to a `Bin` for the purpose of creating compatibility.

The primary `Id` of the `Bin` is determined by the provider. The new `Id` performs as an alias to the primary `Id`. If the alias is a pointer to another bin, it is reassigned to the given bin `Id`.

Parameters

- **`bin_id`** (`osid.id.Id`) – the `Id` of a `Bin`
- **`alias_id`** (`osid.id.Id`) – the alias `Id`

Raise `AlreadyExists` – `alias_id` is already assigned

Raise `NotFound` – `bin_id` not found

Raise `NullArgument` – `bin_id` or `alias_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Bin Hierarchy Methods

`ResourceManager.`**`bin_hierarchy_id`**

Gets the hierarchy `Id` associated with this session.

Returns the hierarchy `Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`bin_hierarchy`**

Gets the hierarchy associated with this session.

Returns the hierarchy associated with this session

Return type `osid.hierarchy.Hierarchy`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`can_access_bin_hierarchy`**`()`

Tests if this user can perform hierarchy queries.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an an application that may not offer traversal functions to unauthorized users.

Returns `false` if hierarchy traversal methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`use_comparative_bin_view`**`()`

The returns from the bin methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`ResourceManager.`**`use_plenary_bin_view`**`()`

A complete view of the `Bin` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`ResourceManager.`**`root_bin_ids`**

Gets the root bin `Ids` in this hierarchy.

Returns the root bin `Ids`

Return type `osid.id.IdList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`root_bins`**

Gets the root bins in the bin hierarchy.

A node with no parents is an orphan. While all bin `Ids` are known to the hierarchy, an orphan does not appear in the hierarchy unless explicitly added as a root node or child of another node.

Returns the root bins

Return type `osid.resource.BinList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method is must be implemented.

`ResourceManager.`**`has_parent_bins`**(*bin_id*)

Tests if the `Bin` has any parents.

Parameters **`bin_id`** (`osid.id.Id`) – the `Id` of a bin

Returns `true` if the bin has parents, `false` otherwise

Return type `boolean`

Raise `NotFound` – `bin_id` is not found

Raise `NullArgument` – `bin_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`is_parent_of_bin`**(*id_*, *bin_id*)

Tests if an `Id` is a direct parent of a bin.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`bin_id`** (`osid.id.Id`) – the `Id` of a bin

Returns `true` if this `id` is a parent of `bin_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `bin_id` is not found

Raise `NullArgument` – `id` or `bin_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` not found return `false`.

`ResourceManager.`**`get_parent_bin_ids`**(*bin_id*)

Gets the parent `Ids` of the given bin.

Parameters **`bin_id`** (`osid.id.Id`) – the `Id` of a bin

Returns the parent `Ids` of the bin

Return type `osid.id.IdList`

Raise `NotFound` – `bin_id` is not found

Raise `NullArgument` – `bin_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`get_parent_bins`**`(`*bin_id*`)`

Gets the parents of the given bin.

Parameters **`bin_id`** (`osid.id.Id`) – the `Id` to query

Returns the parents of the bin

Return type `osid.resource.BinList`

Raise `NotFound` – `bin_id` not found

Raise `NullArgument` – `bin_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`is_ancestor_of_bin`**`(`*id_*, *bin_id*`)`

Tests if an `Id` is an ancestor of a bin.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`bin_id`** (`osid.id.Id`) – the `Id` of a bin

Returns `true` if this `id` is an ancestor of `bin_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `bin_id` is not found

Raise `NullArgument` – `id` or `bin_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` not found return `false`.

`ResourceManager.`**`has_child_bins`**`(`*bin_id*`)`

Tests if a bin has any children.

Parameters **`bin_id`** (`osid.id.Id`) – the `Id` of a bin

Returns `true` if the `bin_id` has children, `false` otherwise

Return type `boolean`

Raise `NotFound` – `bin_id` not found

Raise `NullArgument` – `bin_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`is_child_of_bin`**`(`*id_*, *bin_id*`)`

Tests if a bin is a direct child of another.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`bin_id`** (`osid.id.Id`) – the `Id` of a bin

Returns `true` if the `id` is a child of `bin_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `bin_id` is not found

Raise `NullArgument` – `id` or `bin_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` not found return `false`.

`ResourceManager.`**`get_child_bin_ids`**(*bin_id*)

Gets the child `Ids` of the given bin.

Parameters **`bin_id`** (`osid.id.Id`) – the `Id` to query

Returns the children of the bin

Return type `osid.id.IdList`

Raise `NotFound` – `bin_id` not found

Raise `NullArgument` – `bin_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`get_child_bins`**(*bin_id*)

Gets the children of the given bin.

Parameters **`bin_id`** (`osid.id.Id`) – the `Id` to query

Returns the children of the bin

Return type `osid.resource.BinList`

Raise `NotFound` – `bin_id` not found

Raise `NullArgument` – `bin_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`is_descendant_of_bin`**(*id_*, *bin_id*)

Tests if an `Id` is a descendant of a bin.

Parameters

- **`id`** (`osid.id.Id`) – an `Id`
- **`bin_id`** (`osid.id.Id`) – the `Id` of a bin

Returns `true` if the `id` is a descendant of the `bin_id, false` otherwise

Return type `boolean`

Raise `NotFound` – `bin_id` is not found

Raise `NullArgument` – `id` or `bin_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented. implementation notes: If `id` is not found return `false`.

`ResourceManager.`**`get_bin_node_ids`**(*bin_id*, *ancestor_levels*, *descendant_levels*, *include_siblings*)

Gets a portion of the hierarchy for the given bin.

Parameters

- **`bin_id`** (`osid.id.Id`) – the `Id` to query
- **`ancestor_levels`** (`cardinal`) – the maximum number of ancestor levels to include. A value of 0 returns no parents in the node.
- **`descendant_levels`** (`cardinal`) – the maximum number of descendant levels to include. A value of 0 returns no children in the node.
- **`include_siblings`** (`boolean`) – `true` to include the siblings of the given node, `false` to omit the siblings

Returns a bin node

Return type `osid.hierarchy.Node`

Raise `NotFound` – `bin_id` not found

Raise `NullArgument` – `bin_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`get_bin_nodes`**(*bin_id*, *ancestor_levels*, *descendant_levels*, *include_siblings*)

Gets a portion of the hierarchy for the given bin.

Parameters

- **`bin_id`** (`osid.id.Id`) – the `Id` to query
- **`ancestor_levels`** (`cardinal`) – the maximum number of ancestor levels to include. A value of 0 returns no parents in the node.
- **`descendant_levels`** (`cardinal`) – the maximum number of descendant levels to include. A value of 0 returns no children in the node.
- **`include_siblings`** (`boolean`) – `true` to include the siblings of the given node, `false` to omit the siblings

Returns a bin node

Return type `osid.resource.BinNode`

Raise `NotFound` – `bin_id` not found

Raise `NullArgument` – `bin_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Bin Hierarchy Design Methods

ResourceManager.**bin_hierarchy_id**

Gets the hierarchy `Id` associated with this session.

Returns the hierarchy `Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

ResourceManager.**bin_hierarchy**

Gets the hierarchy associated with this session.

Returns the hierarchy associated with this session

Return type `osid.hierarchy.Hierarchy`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

ResourceManager.**can_modify_bin_hierarchy**()

Tests if this user can change the hierarchy.

A return of true does not guarantee successful authorization. A return of false indicates that it is known performing any update will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer these operations to an unauthorized user.

Returns `false` if changing this hierarchy is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

ResourceManager.**add_root_bin**(*bin_id*)

Adds a root bin.

Parameters **bin_id** (`osid.id.Id`) – the `Id` of a bin

Raise `AlreadyExists` – `bin_id` is already in hierarchy

Raise `NotFound` – `bin_id` not found

Raise `NullArgument` – `bin_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

ResourceManager.**remove_root_bin**(*bin_id*)

Removes a root bin.

Parameters **bin_id** (`osid.id.Id`) – the `Id` of a bin

Raise `NotFound` – `bin_id` not a root

Raise `NullArgument` – `bin_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`add_child_bin`**(*bin_id*, *child_id*)

Adds a child to a bin.

Parameters

- **`bin_id`** (`osid.id.Id`) – the `Id` of a bin
- **`child_id`** (`osid.id.Id`) – the `Id` of the new child

Raise `AlreadyExists` – `bin_id` is already a parent of `child_id`

Raise `NotFound` – `bin_id` or `child_id` not found

Raise `NullArgument` – `bin_id` or `child_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`remove_child_bin`**(*bin_id*, *child_id*)

Removes a child from a bin.

Parameters

- **`bin_id`** (`osid.id.Id`) – the `Id` of a bin
- **`child_id`** (`osid.id.Id`) – the `Id` of the new child

Raise `NotFound` – `bin_id` not a parent of `child_id`

Raise `NullArgument` – `bin_id` or `child_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`ResourceManager.`**`remove_child_bins`**(*bin_id*)

Removes all children from a bin.

Parameters **`bin_id`** (`osid.id.Id`) – the `Id` of a bin

Raise `NotFound` – `bin_id` not in hierarchy

Raise `NullArgument` – `bin_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Bin

Bin

class `dlkit.services.resource.`**`Bin`**(*provider_manager*, *catalog*, *runtime*, *proxy*, ***kwargs*)

Bases: *`dlkit.osid.objects.OsidCatalog`*, `dlkit.osid.sessions.OsidSession`

An inventory defines a collection of resources.

get_bin_record(*bin_record_type*)

Gets the bin record corresponding to the given `Bin` record `Type`.

This method is used to retrieve an object implementing the requested record. The `bin_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(bin_record_type)` is `true` .

Parameters **bin_record_type** (`osid.type.Type`) – the bin record type

Returns the bin record

Return type `osid.resource.records.BinRecord`

Raise `NullArgument` – `bin_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(bin_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Resource Lookup Methods

`Bin.`**bin_id**

Gets the `Bin Id` associated with this session.

Returns the `Bin Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Bin.`**bin**

Gets the `Bin` associated with this session.

Returns the `Bin` associated with this session

Return type `osid.resource.Bin`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bin.`**can_lookup_resources**()

Tests if this user can perform `Resource` lookups.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations.

Returns `false` if lookup methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bin.`**use_comparative_resource_view**()

The returns from the lookup methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`Bin.`**`use_plenary_resource_view`**`()`

A complete view of the `Resource` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`Bin.`**`use_federated_bin_view`**`()`

Federates the view for methods in this session.

A federated view will include resources in bins which are children of this bin in the bin hierarchy.

compliance: mandatory – This method is must be implemented.

`Bin.`**`use_isolated_bin_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts lookups to this bin only.

compliance: mandatory – This method is must be implemented.

`Bin.`**`get_resource`**`(`*resource_id*`)`

Gets the `Resource` specified by its `Id`.

In plenary mode, the exact `Id` is found or a `NotFound` results. Otherwise, the returned `Resource` may have a different `Id` than requested, such as the case where a duplicate `Id` was assigned to a `Resource` and retained for compatibility.

Parameters **`resource_id`** (`osid.id.Id`) – the `Id` of the `Resource` to retrieve

Returns the returned `Resource`

Return type `osid.resource.Resource`

Raise `NotFound` – no `Resource` found with the given `Id`

Raise `NullArgument` – `resource_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bin.`**`get_resources_by_ids`**`(`*resource_ids*`)`

Gets a `ResourceList` corresponding to the given `IdList`.

In plenary mode, the returned list contains all of the resources specified in the `Id` list, in the order of the list, including duplicates, or an error results if an `Id` in the supplied list is not found or inaccessible. Otherwise, inaccessible `Resources` may be omitted from the list and may present the elements in any order including returning a unique set.

Parameters **`resource_ids`** (`osid.id.IdList`) – the list of `Ids` to retrieve

Returns the returned `Resource` list

Return type `osid.resource.ResourceList`

Raise `NotFound` – an `Id was` not found

Raise `NullArgument` – `resource_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Bin.**get_resources_by_genus_type**(*resource_genus_type*)

Gets a ResourceList corresponding to the given resource genus Type which does not include resources of types derived from the specified Type.

In plenary mode, the returned list contains all known resources or an error results. Otherwise, the returned list may contain only those resources that are accessible through this session.

Parameters **resource_genus_type** (osid.type.Type) – a resource genus type

Returns the returned Resource list

Return type osid.resource.ResourceList

Raise NullArgument – resource_genus_type is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Bin.**get_resources_by_parent_genus_type**(*resource_genus_type*)

Gets a ResourceList corresponding to the given resource genus Type and include any additional resources with genus types derived from the specified Type.

In plenary mode, the returned list contains all known resources or an error results. Otherwise, the returned list may contain only those resources that are accessible through this session.

Parameters **resource_genus_type** (osid.type.Type) – a resource genus type

Returns the returned Resource list

Return type osid.resource.ResourceList

Raise NullArgument – resource_genus_type is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Bin.**get_resources_by_record_type**(*resource_record_type*)

Gets a ResourceList containing the given resource record Type.

In plenary mode, the returned list contains all known resources or an error results. Otherwise, the returned list may contain only those resources that are accessible through this session.

Parameters **resource_record_type** (osid.type.Type) – a resource record type

Returns the returned Resource list

Return type osid.resource.ResourceList

Raise NullArgument – resource_record_type is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Bin.**resources**

Gets all Resources.

In plenary mode, the returned list contains all known resources or an error results. Otherwise, the returned list may contain only those resources that are accessible through this session.

Returns a list of `Resources`

Return type `osid.resource.ResourceList`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Resource Query Methods

`Bin.`**`bin_id`**

Gets the `Bin Id` associated with this session.

Returns the `Bin Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Bin.`**`bin`**

Gets the `Bin` associated with this session.

Returns the `Bin` associated with this session

Return type `osid.resource.Bin`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bin.`**`can_search_resources`**`()`

Tests if this user can perform `Resource` searches.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer search operations to unauthorized users.

Returns `false` if search methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bin.`**`use_federated_bin_view`**`()`

Federates the view for methods in this session.

A federated view will include resources in bins which are children of this bin in the bin hierarchy.

compliance: mandatory – This method is must be implemented.

`Bin.`**`use_isolated_bin_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts lookups to this bin only.

compliance: mandatory – This method is must be implemented.

`Bin.`**`resource_query`**

Gets a resource query.

The returned query will not have an extension query.

Returns the resource query

Return type `osid.resource.ResourceQuery`

compliance: mandatory – This method must be implemented.

`Bin.`**`get_resources_by_query`**(*resource_query*)

Gets a list of `Resources` matching the given resource query.

Parameters **`resource_query`** (`osid.resource.ResourceQuery`) – the resource query

Returns the returned `ResourceList`

Return type `osid.resource.ResourceList`

Raise `NullArgument` – `resource_query` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `resource_query` is not of this service

compliance: mandatory – This method must be implemented.

Resource Search Methods

`Bin.`**`resource_search`**

Gets a resource search.

Returns the resource search

Return type `osid.resource.ResourceSearch`

compliance: mandatory – This method must be implemented.

`Bin.`**`resource_search_order`**

Gets a resource search order.

The `ResourceSearchOrder` is supplied to a `ResourceSearch` to specify the ordering of results.

Returns the resource search order

Return type `osid.resource.ResourceSearchOrder`

compliance: mandatory – This method must be implemented.

`Bin.`**`get_resources_by_search`**(*resource_query*, *resource_search*)

Gets the search results matching the given search query using the given search.

Parameters

- **`resource_query`** (`osid.resource.ResourceQuery`) – the resource query
- **`resource_search`** (`osid.resource.ResourceSearch`) – the resource search

Returns the resource search results

Return type `osid.resource.ResourceSearchResults`

Raise `NullArgument` – `resource_query` or `resource_search` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `resource_query` or `resource_search` is not of this service

compliance: mandatory – This method must be implemented.

`Bin.`**`get_resource_query_from_inspector`**`(`*resource_query_inspector*`)`

Gets a resource query from an inspector.

The inspector is available from a `ResourceSearchResults`.

Parameters **`resource_query_inspector`** (`osid.resource.ResourceQueryInspector`) – a resource query inspector

Returns the resource query

Return type `osid.resource.ResourceQuery`

Raise `NullArgument` – `resource_query_inspector` is `null`

Raise `Unsupported` – `resource_query_inspector` is not of this service

compliance: mandatory – This method must be implemented.

Resource Admin Methods

`Bin.`**`bin_id`**

Gets the `Bin Id` associated with this session.

Returns the `Bin Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Bin.`**`bin`**

Gets the `Bin` associated with this session.

Returns the `Bin` associated with this session

Return type `osid.resource.Bin`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bin.`**`can_create_resources`**`()`

Tests if this user can create `Resources`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known creating a `Resource` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer create operations to an unauthorized user.

Returns `false` if `Resource` creation is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bin.`**`can_create_resource_with_record_types`**`(`*resource_record_types*`)`

Tests if this user can create a single `Resource` using the desired record types.

While `ResourceManager.getResourceRecordTypes()` can be used to examine which records are supported, this method tests which record(s) are required for creating a specific `Resource`. Providing an empty array tests if a `Resource` can be created with no records.

Parameters **`resource_record_types`** (`osid.type.Type[]`) – array of resource record types

Returns `true` if `Resource` creation using the specified `Types` is supported, `false` otherwise

Return type `boolean`

Raise `NullArgument` – `resource_record_types` is `null`

compliance: mandatory – This method must be implemented.

`Bin.`**`get_resource_form_for_create`**(*resource_record_types*)

Gets the resource form for creating new resources.

A new form should be requested for each create transaction.

Parameters **`resource_record_types`** (`osid.type.Type[]`) – array of resource record types

Returns the resource form

Return type `osid.resource.ResourceForm`

Raise `NullArgument` – `resource_record_types` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – unable to get form with requested record types

compliance: mandatory – This method must be implemented.

`Bin.`**`create_resource`**(*resource_form*)

Creates a new `Resource`.

Parameters **`resource_form`** (`osid.resource.ResourceForm`) – the form for this `Resource`

Returns the new `Resource`

Return type `osid.resource.Resource`

Raise `IllegalState` – `resource_form` already used in a create transaction

Raise `InvalidArgument` – one or more of the form elements is invalid

Raise `NullArgument` – `resource_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `resource_form` did not originate from `get_resource_form_for_create()`

compliance: mandatory – This method must be implemented.

`Bin.`**`can_update_resources`**()

Tests if this user can update `Resources`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known updating a `Resource` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer update operations to an unauthorized user.

Returns `false` if `Resource` modification is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bin.`**`get_resource_form_for_update`**`(`*resource_id*`)`
Gets the resource form for updating an existing resource.

A new resource form should be requested for each update transaction.

Parameters **`resource_id`** (`osid.id.Id`) – the `Id` of the `Resource`

Returns the resource form

Return type `osid.resource.ResourceForm`

Raise `NotFound` – `resource_id` is not found

Raise `NullArgument` – `resource_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bin.`**`update_resource`**`(`*resource_form*`)`
Updates an existing resource.

Parameters **`resource_form`** (`osid.resource.ResourceForm`) – the form containing the elements to be updated

Raise `IllegalState` – `resource_form` already used in an update transaction

Raise `InvalidArgument` – the form contains an invalid value

Raise `NullArgument` – `resource_form` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

Raise `Unsupported` – `resource_form` did not originate from `get_resource_form_for_update()`

compliance: mandatory – This method must be implemented.

`Bin.`**`can_delete_resources`**`()`
Tests if this user can delete `Resources`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting a `Resource` will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer delete operations to an unauthorized user.

Returns `false` if `Resource` deletion is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bin.`**`delete_resource`**`(`*resource_id*`)`
Deletes a `Resource`.

Parameters **`resource_id`** (`osid.id.Id`) – the `Id` of the `Resource` to remove

Raise `NotFound` – `resource_id` not found

Raise `NullArgument` – `resource_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bin.`**`can_manage_resource_aliases`**`()`

Tests if this user can manage `Id` aliases for `Resources`.

A return of true does not guarantee successful authorization. A return of false indicates that it is known changing an alias will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer alias operations to an unauthorized user.

Returns `false` if `Resource` aliasing is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bin.`**`alias_resource`**`(`*resource_id*, *alias_id*`)`

Adds an `Id` to a `Resource` for the purpose of creating compatibility.

The primary `Id` of the `Resource` is determined by the provider. The new `Id` performs as an alias to the primary `Id`. If the alias is a pointer to another resource it is reassigned to the given resource `Id`.

Parameters

- **`resource_id`** (`osid.id.Id`) – the `Id` of a `Resource`
- **`alias_id`** (`osid.id.Id`) – the alias `Id`

Raise `AlreadyExists` – `alias_id` is already assigned

Raise `NotFound` – `resource_id` not found

Raise `NullArgument` – `alias_id` or `resource_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Resource Notification Methods

`Bin.`**`bin_id`**

Gets the `Bin Id` associated with this session.

Returns the `Bin Id` associated with this session

Return type `osid.id.Id`

compliance: mandatory – This method must be implemented.

`Bin.`**`bin`**

Gets the `Bin` associated with this session.

Returns the `Bin` associated with this session

Return type `osid.resource.Bin`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bin.`**`can_register_for_resource_notifications`**`()`

Tests if this user can register for `Resource` notifications.

A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer notification operations.

Returns `false` if notification methods are not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bin.`**`use_federated_bin_view`**`()`

Federates the view for methods in this session.

A federated view will include resources in bins which are children of this bin in the bin hierarchy.

compliance: mandatory – This method is must be implemented.

`Bin.`**`use_isolated_bin_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts lookups to this bin only.

compliance: mandatory – This method is must be implemented.

`Bin.`**`register_for_new_resources`**`()`

Register for notifications of new resources.

`ResourceReceiver.newResources()` is invoked when a new `Resource` is appears in this bin.

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bin.`**`register_for_changed_resources`**`()`

Registers for notification of updated resources.

`ResourceReceiver.changedResources()` is invoked when a resource in this bin is changed.

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bin.`**`register_for_changed_resource`**`(`*resource_id*`)`

Registers for notification of an updated resource.

`ResourceReceiver.changedResources()` is invoked when the specified resource in this bin is changed.

Parameters **`resource_id`** (`osid.id.Id`) – the `Id` of the `Resource` to monitor

Raise `NullArgument` – `resource_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Bin.**register_for_deleted_resources**()

Registers for notification of deleted resources.

`ResourceReceiver.deletedResources()` is invoked when a resource is deleted or removed from this bin.

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Bin.**register_for_deleted_resource**(*resource_id*)

Registers for notification of a deleted resource.

`ResourceReceiver.deletedResources()` is invoked when the specified resource is deleted or removed from this bin.

Parameters **resource_id** (`osid.id.Id`) – the `Id` of the `Resource` to monitor

Raise `NullArgument` – `resource_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Bin.**reliable_resource_notifications**()

Reliable notifications are desired.

In reliable mode, notifications are to be acknowledged using `acknowledge_item_notification()` .

compliance: mandatory – This method is must be implemented.

Bin.**unreliable_resource_notifications**()

Unreliable notifications are desired.

In unreliable mode, notifications do not need to be acknowledged.

compliance: mandatory – This method is must be implemented.

Bin.**acknowledge_resource_notification**(*notification_id*)

Acknowledge an resource notification.

Parameters **notification_id** (`osid.id.Id`) – the `Id` of the notification

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Resource Bin Methods

Bin.**use_comparative_bin_view**()

The returns from the lookup methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

Bin.**use_plenary_bin_view**()

A complete view of the `Resource` and `Bin` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

Bin.**can_lookup_resource_bin_mappings**()

Tests if this user can perform lookups of resource/bin mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known lookup methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns `false` if looking up mappings is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

Bin.**get_resource_ids_by_bin**(*bin_id*)

Gets the list of `Resource Ids` associated with a `Bin`.

Parameters **bin_id** (`osid.id.Id`) – `Id` of a `Bin`

Returns list of related resource `Ids`

Return type `osid.id.IdList`

Raise `NotFound` – `bin_id` is not found

Raise `NullArgument` – `bin_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Bin.**get_resources_by_bin**(*bin_id*)

Gets the list of `Resources` associated with a `Bin`.

Parameters **bin_id** (`osid.id.Id`) – `Id` of a `Bin`

Returns list of related resources

Return type `osid.resource.ResourceList`

Raise `NotFound` – `bin_id` is not found

Raise `NullArgument` – `bin_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Bin.**get_resource_ids_by_bins**(*bin_ids*)

Gets the list of `Resource Ids` corresponding to a list of `Bin` objects.

Parameters **bin_ids** (`osid.id.IdList`) – list of bin `Ids`

Returns list of resource `Ids`

Return type `osid.id.IdList`

Raise `NullArgument` – `bin_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bin.`**`get_resources_by_bins`**`(`*bin_ids*`)`

Gets the list of `Resources` corresponding to a list of `Bins`.

Parameters **`bin_ids`** (`osid.id.IdList`) – list of bin `Ids`

Returns list of resources

Return type `osid.resource.ResourceList`

Raise `NullArgument` – `bin_ids` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bin.`**`get_bin_ids_by_resource`**`(`*resource_id*`)`

Gets the list of `Bin Ids` mapped to a `Resource`.

Parameters **`resource_id`** (`osid.id.Id`) – `Id` of a `Resource`

Returns list of bin `Ids`

Return type `osid.id.IdList`

Raise `NotFound` – `resource_id` is not found

Raise `NullArgument` – `resource_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bin.`**`get_bins_by_resource`**`(`*resource_id*`)`

Gets the list of `Bin` objects mapped to a `Resource`.

Parameters **`resource_id`** (`osid.id.Id`) – `Id` of a `Resource`

Returns list of bins

Return type `osid.resource.BinList`

Raise `NotFound` – `resource_id` is not found

Raise `NullArgument` – `resource_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Resource Bin Assignment Methods

`Bin.`**`can_assign_resources`**`()`

Tests if this user can alter resource/bin mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known mapping methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer assignment operations to unauthorized users.

Returns `false` if mapping is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bin.`**`can_assign_resources_to_bin`**(*bin_id*)

Tests if this user can alter resource/bin mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known mapping methods in this session will result in a `PermissionDenied` . This is intended as a hint to an application that may opt not to offer assignment operations to unauthorized users.

Parameters **`bin_id`** (`osid.id.Id`) – the `Id` of the `Bin`

Returns `false` if mapping is not authorized, `true` otherwise

Return type `boolean`

Raise `NullArgument` – `bin_id` is `null`

compliance: mandatory – This method must be implemented.

`Bin.`**`get_assignable_bin_ids`**(*bin_id*)

Gets a list of bins including and under the given bin node in which any resource can be assigned.

Parameters **`bin_id`** (`osid.id.Id`) – the `Id` of the `Bin`

Returns list of assignable bin `Ids`

Return type `osid.id.IdList`

Raise `NullArgument` – `bin_id` is `null`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`Bin.`**`get_assignable_bin_ids_for_resource`**(*bin_id*, *resource_id*)

Gets a list of bins including and under the given bin node in which a specific resource can be assigned.

Parameters

- **`bin_id`** (`osid.id.Id`) – the `Id` of the `Bin`
- **`resource_id`** (`osid.id.Id`) – the `Id` of the `Resource`

Returns list of assignable bin `Ids`

Return type `osid.id.IdList`

Raise `NullArgument` – `bin_id` or `resource_id` is `null`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`Bin.`**`assign_resource_to_bin`**(*resource_id*, *bin_id*)

Adds an existing `Resource` to a `Bin`.

Parameters

- **`resource_id`** (`osid.id.Id`) – the `Id` of the `Resource`

- **bin_id** (osid.id.Id) – the Id of the Bin

Raise AlreadyExists – resource_id is already assigned to bin_id

Raise NotFound – resource_id or bin_id not found

Raise NullArgument – resource_id or bin_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Bin.**unassign_resource_from_bin**(*resource_id*, *bin_id*)

Removes a Resource from a Bin.

Parameters

- **resource_id** (osid.id.Id) – the Id of the Resource
- **bin_id** (osid.id.Id) – the Id of the Bin

Raise NotFound – resource_id or bin_id not found or resource_id not assigned to bin_id

Raise NullArgument – resource_id or bin_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Resource Agent Methods

Bin.**bin_id**

Gets the Bin Id associated with this session.

Returns the Bin Id associated with this session

Return type osid.id.Id

compliance: mandatory – This method must be implemented.

Bin.**bin**

Gets the Bin associated with this session.

Returns the Bin associated with this session

Return type osid.resource.Bin

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Bin.**can_lookup_resource_agent_mappings**()

Tests if this user can perform lookups of resource/agent mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known lookup methods in this session will result in a PermissionDenied. This is intended as a hint to an application that may opt not to offer lookup operations to unauthorized users.

Returns false if looking up mappings is not authorized, true otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`Bin.`**`use_comparative_agent_view`**`()`

The returns from the lookup methods may omit or translate elements based on this session, such as authorization, and not result in an error.

This view is used when greater interoperability is desired at the expense of precision.

compliance: mandatory – This method is must be implemented.

`Bin.`**`use_plenary_agent_view`**`()`

A complete view of the `Agent` returns is desired.

Methods will return what is requested or result in an error. This view is used when greater precision is desired at the expense of interoperability.

compliance: mandatory – This method is must be implemented.

`Bin.`**`use_federated_bin_view`**`()`

Federates the view for methods in this session.

A federated view will include resources in bins which are children of this bin in the bin hierarchy.

compliance: mandatory – This method is must be implemented.

`Bin.`**`use_isolated_bin_view`**`()`

Isolates the view for methods in this session.

An isolated view restricts lookups to this bin only.

compliance: mandatory – This method is must be implemented.

`Bin.`**`get_resource_id_by_agent`**`(`*agent_id*`)`

Gets the `Resource Id` associated with the given agent.

Parameters **`agent_id`** (`osid.id.Id`) – `Id` of the `Agent`

Returns associated resource

Return type `osid.id.Id`

Raise `NotFound` – `agent_id` is not found

Raise `NullArgument` – `agent_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

`Bin.`**`get_resource_by_agent`**`(`*agent_id*`)`

Gets the `Resource` associated with the given agent.

Parameters **`agent_id`** (`osid.id.Id`) – `Id` of the `Agent`

Returns associated resource

Return type `osid.resource.Resource`

Raise `NotFound` – `agent_id` is not found

Raise `NullArgument` – `agent_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Bin.**get_agent_ids_by_resource**(*resource_id*)

Gets the list of Agent Ids mapped to a Resource.

Parameters **resource_id** (osid.id.Id) – Id of a Resource

Returns list of agent Ids

Return type osid.id.IdList

Raise NotFound – resource_id is not found

Raise NullArgument – resource_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Bin.**get_agents_by_resource**(*resource_id*)

Gets the list of Agents mapped to a Resource.

Parameters **resource_id** (osid.id.Id) – Id of a Resource

Returns list of agents

Return type osid.authentication.AgentList

Raise NotFound – resource_id is not found

Raise NullArgument – resource_id is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Resource Agent Assignment Methods

Bin.**bin_id**

Gets the Bin Id associated with this session.

Returns the Bin Id associated with this session

Return type osid.id.Id

compliance: mandatory – This method must be implemented.

Bin.**bin**

Gets the Bin associated with this session.

Returns the Bin associated with this session

Return type osid.resource.Bin

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

compliance: mandatory – This method must be implemented.

Bin.**can_assign_agents**()

Tests if this user can alter resource/agent mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known mapping methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer assignment operations to unauthorized users.

Returns `false` if mapping is not authorized, `true` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

Bin.**can_assign_agents_to_resource**(*resource_id*)

Tests if this user can alter resource/agent mappings.

A return of true does not guarantee successful authorization. A return of false indicates that it is known location methods in this session will result in a `PermissionDenied`. This is intended as a hint to an application that may opt not to offer assignment operations to unauthorized users.

Parameters **resource_id** (`osid.id.Id`) – the `Id` of the `Resource`

Returns `false` if mapping is not authorized, `true` otherwise

Return type `boolean`

Raise `NullArgument` – `resource_id` is `null`

compliance: mandatory – This method must be implemented.

Bin.**assign_agent_to_resource**(*agent_id*, *resource_id*)

Adds an existing `Agent` to a `Resource`.

Parameters

- **agent_id** (`osid.id.Id`) – the `Id` of the `Agent`
- **resource_id** (`osid.id.Id`) – the `Id` of the `Resource`

Raise `AlreadyExists` – `agent_id` is already assigned to `resource_id`

Raise `NotFound` – `agent_id` or `resource_id` not found

Raise `NullArgument` – `agent_id` or `resource_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Bin.**unassign_agent_from_resource**(*agent_id*, *resource_id*)

Removes an `Agent` from a `Resource`.

Parameters

- **agent_id** (`osid.id.Id`) – the `Id` of the `Agent`
- **resource_id** (`osid.id.Id`) – the `Id` of the `Resource`

Raise `NotFound` – `agent_id` or `resource_id` not found or `agent_id` not assigned to `resource_id`

Raise `NullArgument` – `agent_id` or `resource_id` is `null`

Raise `OperationFailed` – unable to complete request

Raise `PermissionDenied` – authorization failure

compliance: mandatory – This method must be implemented.

Objects

Resource

class `dlkit.resource.objects.`**`Resource`**

Bases: *`dlkit.osid.objects.OsidObject`*

A `Resource` represents an arbitrary entity.

Resources are used to define an object to accompany an OSID `Id` used in other OSIDs. A resource may be used to represent a meeting room in the Scheduling OSID, or a student in the Course OSID.

A `Resource` may also represent a group or organization. A provider may present such a group in an opaque manner through a single resource definition, or the provider may expose the resource collection for examination or manipulation. If such a resource collection is visible, `is_group()` is `true` and can be used in one of the group sessions available in this OSID.

`is_group``()`

Tests if this resource is a group.

A resource that is a group can be used in the group sessions.

Returns `true` if this resource is a group, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`is_demographic``()`

Tests if this resource is a demographic.

A resource that is a demographic can be used in the demographic service and the group sessions.

Returns `true` if this resource is a demographic, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`has_avatar``()`

Tests if this resource has an avatar.

Returns `true` if this resource has an avatar, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`avatar_id`

Gets the asset `Id`.

Returns the asset `Id`

Return type `osid.id.Id`

Raise `IllegalState` – `has_avatar()` is `false`

compliance: mandatory – This method must be implemented.

`avatar`

Gets the asset.

Returns the asset

Return type `osid.repository.Asset`

Raise `IllegalState` – `has_avatar()` is `false`

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

get_resource_record(*resource_record_type*)

Gets the resource record corresponding to the given `Resource` record `Type`.

This method is used to retrieve an object implementing the requested record. The `resource_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(resource_record_type)` is `true` .

Parameters **resource_record_type** (`osid.type.Type`) – the resource record type

Returns the resource record

Return type `osid.resource.records.ResourceRecord`

Raise `NullArgument` – `resource_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(resource_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Resource Form

class `dlkit.resource.objects.`**ResourceForm**

Bases: *`dlkit.osid.objects.OsidObjectForm`*

This is the form for creating and updating `Resources`.

Like all `OsidForm` objects, various data elements may be set here for use in the create and update methods in the `ResourceAdminSession`. For each data element that may be set, metadata may be examined to provide display hints or data constraints.

Resources can be designated as a group. The group metadata indicates if it is possible to convert a resource to a group and vice-versa.

group_metadata

Gets the metadata for a group.

Returns metadata for the group

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

group

avatar_metadata

Gets the metadata for an asset.

Returns metadata for the asset

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

avatar

get_resource_form_record(*resource_record_type*)

Gets the `ResourceFormRecord` corresponding to the given `Resource` record `Type`.

Parameters **resource_record_type** (`osid.type.Type`) – the resource record type

Returns the resource form record

Return type `osid.resource.records.ResourceFormRecord`

Raise `NullArgument` – `resource_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(resource_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Resource List

class `dlkit.resource.objects.`**ResourceList**

Bases: *dlkit.osid.objects.OsidList*

Like all `OsidLists, ResourceList` provides a means for accessing `Resource` elements sequentially either one at a time or many at a time.

Examples: while (rl.hasNext()) { Resource resource = rl.getNextResource(); }

or

while (rl.hasNext()) { Resource[] resources = rl.getNextResources(rl.available());

}

next_resource

Gets the next `Resource` in this list.

Returns the next `Resource` in this list. The `has_next()` method should be used to test that a next `Resource` is available before calling this method.

Return type `osid.resource.Resource`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

get_next_resources(*n*)

Gets the next set of `Resources` in this list which must be less than or equal to the return from `available()`.

Parameters **n** (`cardinal`) – the number of `Resource` elements requested which must be less than or equal to `available()`

Returns an array of `Resource` elements.The length of the array is less than or equal to the number specified.

Return type `osid.resource.Resource`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Resource Node

class `dlkit.resource.objects.`**`ResourceNode`**

Bases: *`dlkit.osid.objects.OsidNode`*

This interface is a container for a partial hierarchy retrieval.

The number of hierarchy levels traversable through this interface depend on the number of levels requested in the `BinHierarchySession`.

`resource`

Gets the `Resource` at this node.

Returns the resource represented by this node

Return type `osid.resource.Resource`

compliance: mandatory – This method must be implemented.

`parent_resource_nodes`

Gets the parents of this resource.

Returns the parents of the resource

Return type `osid.resource.ResourceNodeList`

compliance: mandatory – This method must be implemented.

`child_resource_nodes`

Gets the children of this resource.

Returns the children of this resource

Return type `osid.resource.ResourceNodeList`

compliance: mandatory – This method must be implemented.

Resource Node List

class `dlkit.resource.objects.`**`ResourceNodeList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists`, `ResourceNodeList` provides a means for accessing `ResourceNode` elements sequentially either one at a time or many at a time.

Examples: while (rnl.hasNext()) { ResourceNode node = rnl.getNextResourceNode(); }

or

while rnl.hasNext()) { ResourceNode[] nodes = rnl.getNextResourceNodes(rnl.available());

}

`next_resource_node`

Gets the next `ResourceNode` in this list.

Returns the next `ResourceNode` in this list. The `has_next()` method should be used to test that a next `ResourceNode` is available before calling this method.

Return type `osid.resource.ResourceNode`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

get_next_resource_nodes(*n*)
Gets the next set of `ResourceNode` elements in this list which must be less than or equal to the return from `available()`.

Parameters **n** (`cardinal`) – the number of `ResourceNode` elements requested which must be less than or equal to `available()`

Returns an array of `ResourceNode` elements.The length of the array is less than or equal to the number specified.

Return type `osid.resource.ResourceNode`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Bin

class `dlkit.resource.objects.`**Bin**(*abc_resource_objects.Bin*, *osid_objects.OsidCatalog*)
:noindex:

get_bin_record(*bin_record_type*)
Gets the bin record corresponding to the given `Bin` record `Type`.

This method is used to retrieve an object implementing the requested record. The `bin_record_type` may be the `Type` returned in `get_record_types()` or any of its parents in a `Type` hierarchy where `has_record_type(bin_record_type)` is `true` .

Parameters **bin_record_type** (`osid.type.Type`) – the bin record type

Returns the bin record

Return type `osid.resource.records.BinRecord`

Raise `NullArgument` – `bin_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(bin_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Bin Form

class `dlkit.resource.objects.`**BinForm**
Bases: *dlkit.osid.objects.OsidCatalogForm*

This is the form for creating and updating bins.

Like all `OsidForm` objects, various data elements may be set here for use in the create and update methods in the `BinAdminSession`. For each data element that may be set, metadata may be examined to provide display hints or data constraints.

get_bin_form_record(*bin_record_type*)
Gets the `BinFormRecord` corresponding to the given bin record `Type`.

Parameters **bin_record_type** (`osid.type.Type`) – the bin record type

Returns the bin form record

Return type `osid.resource.records.BinFormRecord`

Raise `NullArgument` – `bin_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(bin_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Bin List

class `dlkit.resource.objects.`**`BinList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists, BinList` provides a means for accessing `Bin` elements sequentially either one at a time or many at a time.

Examples: while (bl.hasNext()) { Bin bin = bl.getNextBin(); }

or

while (bl.hasNext()) { Bin[] bins = bl.getNextBins(bl.available());

}

`next_bin`

Gets the next `Bin` in this list.

Returns the next `Bin` in this list. The `has_next()` method should be used to test that a next `Bin` is available before calling this method.

Return type `osid.resource.Bin`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_bins`(*n*)

Gets the next set of `Bin` elements in this list which must be less than or equal to the return from `available()`.

Parameters **`n`** (`cardinal`) – the number of `Bin` elements requested which must be less than or equal to `available()`

Returns an array of `Bin` elements.The length of the array is less than or equal to the number specified.

Return type `osid.resource.Bin`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Bin Node

class `dlkit.resource.objects.`**`BinNode`**

Bases: *`dlkit.osid.objects.OsidNode`*

This interface is a container for a partial hierarchy retrieval.

The number of hierarchy levels traversable through this interface depend on the number of levels requested in the `BinHierarchySession`.

`bin`

Gets the `Bin` at this node.

Returns the bin represented by this node

Return type `osid.resource.Bin`

compliance: mandatory – This method must be implemented.

`parent_bin_nodes`

Gets the parents of this bin.

Returns the parents of the `id`

Return type `osid.resource.BinNodeList`

compliance: mandatory – This method must be implemented.

`child_bin_nodes`

Gets the children of this bin.

Returns the children of this bin

Return type `osid.resource.BinNodeList`

compliance: mandatory – This method must be implemented.

Bin Node List

class `dlkit.resource.objects.`**`BinNodeList`**

Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists, BinNodeList` provides a means for accessing `BinNode` elements sequentially either one at a time or many at a time.

Examples: while (bnl.hasNext()) { BinNode node = bnl.getNextBinNode(); }

or

while (bnl.hasNext()) { BinNode[] nodes = bnl.getNextBinNodes(bnl.available());

}

`next_bin_node`

Gets the next `BinNode` in this list.

Returns the next `BinNode` in this list. The `has_next()` method should be used to test that a next `BinNode` is available before calling this method.

Return type `osid.resource.BinNode`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`get_next_bin_nodes``(`*n*`)`

Gets the next set of `BinNode` elements in this list which must be less than or equal to the return from `available()`.

Parameters **`n`** (`cardinal`) – the number of `BinNode` elements requested which must be less than or equal to `available()`

Returns an array of `BinNode` elements.The length of the array is less than or equal to the number specified.

Return type `osid.resource.BinNode`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Queries

Resource Query

class `dlkit.resource.queries.`**`ResourceQuery`**

Bases: *`dlkit.osid.queries.OsidObjectQuery`*

This is the query for searching resources.

Each method specifies an `AND` term while multiple invocations of the same method produce a nested `OR`.

`match_group`(*match*)

Matches resources that are also groups.

Parameters **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

compliance: mandatory – This method must be implemented.

`group_terms`

`match_demographic`(*match*)

Matches resources that are also demographics.

Parameters **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

compliance: mandatory – This method must be implemented.

`demographic_terms`

`match_containing_group_id`(*resource_id*, *match*)

Sets the group `Id` for this query to match resources within the given group.

Parameters

- **`resource_id`** (`osid.id.Id`) – a resource `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `resource_id` is `null`

compliance: mandatory – This method must be implemented.

`containing_group_id_terms`

`supports_containing_group_query`()

Tests if a `ResourceQuery` is available for querying containing groups.

Returns `true` if a group resource query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

containing_group_query
Gets the query for a a containing group.

Multiple retrievals produce a nested `OR` term.

Returns the resource query

Return type `osid.resource.ResourceQuery`

Raise `Unimplemented` – `supports_containing_group_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_agent_query()‘‘ is ‘‘true‘‘.

match_any_containing_group(*match*)
Matches resources inside any group.

Parameters **match** (`boolean`) – `true` to match any containing group, `false` to match resources part of no groups

compliance: mandatory – This method must be implemented.

containing_group_terms

match_avatar_id(*asset_id*, *match*)
Sets the asset `Id` for this query.

Parameters

- **asset_id** (`osid.id.Id`) – the asset `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `asset_id` is `null`

compliance: mandatory – This method must be implemented.

avatar_id_terms

supports_avatar_query()
Tests if an `AssetQuery` is available.

Returns `true` if an asset query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

avatar_query
Gets the query for an asset.

Multiple retrievals produce a nested `OR` term.

Returns the asset query

Return type `osid.repository.AssetQuery`

Raise `Unimplemented` – `supports_avatar_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_avatar_query()‘‘ is ‘‘true‘‘.

match_any_avatar(*match*)
Matches resources with any asset.

Parameters **match** (`boolean`) – `true` to match any asset, `false` to match resources with no asset

compliance: mandatory – This method must be implemented.

avatar_terms

match_agent_id(*agent_id*, *match*)
Sets the agent `Id` for this query.

Parameters

- **agent_id** (`osid.id.Id`) – the agent `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `agent_id` is `null`

compliance: mandatory – This method must be implemented.

agent_id_terms

supports_agent_query()
Tests if an `AgentQuery` is available.

Returns `true` if an agent query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

agent_query
Gets the query for an agent.

Multiple retrievals produce a nested `OR` term.

Returns the agent query

Return type `osid.authentication.AgentQuery`

Raise `Unimplemented` – `supports_agent_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_agent_query()‘‘ is ‘‘true‘‘.

match_any_agent(*match*)
Matches resources with any agent.

Parameters **match** (`boolean`) – `true` to match any agent, `false` to match resources with no agent

compliance: mandatory – This method must be implemented.

agent_terms

match_resource_relationship_id(*resource_relationship_id*, *match*)
Sets the resource relationship `Id` for this query.

Parameters

- **resource_relationship_id** (`osid.id.Id`) – the resource relationship `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `resource_relationship_id` is `null`

compliance: mandatory – This method must be implemented.

resource_relationship_id_terms

supports_resource_relationship_query()
Tests if a `ResourceRelationshipQuery` is available.

Returns `true` if a resource relationship query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

resource_relationship_query

Gets the query for aa resource relationship.

Multiple retrievals produce a nested `OR` term.

Returns the resource relationship query

Return type `osid.resource.ResourceRelationshipQuery`

Raise `Unimplemented` – `supports_resource_relationship_query()` is `false`

compliance: optional – This method must be implemented if ``supports_resource_relationship_query()`` is ``true``.

match_any_resource_relationship(*match*)

Matches resources with any resource relationship.

Parameters match (`boolean`) – `true` to match any resource relationship, `false` to match resources with no relationship

compliance: mandatory – This method must be implemented.

resource_relationship_terms

match_bin_id(*bin_id*, *match*)

Sets the bin `Id` for this query.

Parameters

- **bin_id** (`osid.id.Id`) – the bin `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `bin_id` is `null`

compliance: mandatory – This method must be implemented.

bin_id_terms

supports_bin_query()

Tests if a `BinQuery` is available.

Returns `true` if a bin query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

bin_query

Gets the query for a bin.

Multiple retrievals produce a nested `OR` term.

Returns the bin query

Return type `osid.resource.BinQuery`

Raise `Unimplemented` – `supports_bin_query()` is `false`

compliance: optional – This method must be implemented if ``supports_bin_query()`` is ``true``.

bin_terms

get_resource_query_record(*resource_record_type*)

Gets the resource query record corresponding to the given `Resource` record `Type`.

Multiple retrievals produce a nested `OR` term.

Parameters **`resource_record_type`** (`osid.type.Type`) – a resource record type

Returns the resource query record

Return type `osid.resource.records.ResourceQueryRecord`

Raise `NullArgument` – `resource_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(resource_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Bin Query

class `dlkit.resource.queries.`**`BinQuery`**

Bases: *`dlkit.osid.queries.OsidCatalogQuery`*

This is the query for searching bins.

Each method specifies an `AND` term while multiple invocations of the same method produce a nested `OR`.

`match_resource_id`(*resource_id*, *match*)

Sets the resource `Id` for this query.

Parameters

- **`resource_id`** (`osid.id.Id`) – a resource `Id`
- **`match`** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `resource_id` is `null`

compliance: mandatory – This method must be implemented.

`resource_id_terms`

`supports_resource_query`()

Tests if a `ResourceQuery` is available.

Returns `true` if a resource query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`resource_query`

Gets the query for a resource.

Multiple retrievals produce a nested `OR` term.

Returns the resource query

Return type `osid.resource.ResourceQuery`

Raise `Unimplemented` – `supports_resource_query()` is `false`

compliance: optional – This method must be implemented if ‘‘supports_resource_query()‘‘ is ‘‘true‘‘.

`match_any_resource`(*match*)

Matches bins with any resource.

Parameters **`match`** (`boolean`) – `true` to match bins with any resource, `false` to match bins with no resources

compliance: mandatory – This method must be implemented.

resource_terms

match_ancestor_bin_id(*binid*, *match*)
Sets the bin `Id` for this query to match bins that have the specified bin as an ancestor.

Parameters

- **binid** (`osid.id.Id`) – a bin `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `bin_id` is `null`

compliance: mandatory – This method must be implemented.

ancestor_bin_id_terms

supports_ancestor_bin_query()
Tests if a `BinQuery` is available.

Returns `true` if a bin query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

ancestor_bin_query
Gets the query for a bin.

Multiple retrievals produce a nested `OR` term.

Returns the bin query

Return type `osid.resource.BinQuery`

Raise `Unimplemented` – `supports_ancestor_bin_query()` is `false`

compliance: optional – This method must be implemented if ``supports_ancestor_bin_query()`` is ``true``.

match_any_ancestor_bin(*match*)
Matches bins with any ancestor.

Parameters **match** (`boolean`) – `true` to match bins with any ancestor, `false` to match root bins

compliance: mandatory – This method must be implemented.

ancestor_bin_terms

match_descendant_bin_id(*binid*, *match*)
Sets the bin `Id` for this query to match bins that have the specified bin as a descendant.

Parameters

- **binid** (`osid.id.Id`) – a bin `Id`
- **match** (`boolean`) – `true` for a positive match, `false` for a negative match

Raise `NullArgument` – `bin_id` is `null`

compliance: mandatory – This method must be implemented.

descendant_bin_id_terms

supports_descendant_bin_query()
Tests if a `BinQuery` is available.

Returns `true` if a bin query is available, `false` otherwise

Return type `boolean`

compliance: mandatory – This method must be implemented.

`descendant_bin_query`

Gets the query for a bin.

Multiple retrievals produce a nested `OR` term.

Returns the bin query

Return type `osid.resource.BinQuery`

Raise `Unimplemented` – `supports_descendant_bin_query()` is `false`

compliance: optional – This method must be implemented if ``supports_descendant_bin_query()`` is ``true``.

`match_any_descendant_bin`(*match*)

Matches bins with any descendant.

Parameters **`match`** (`boolean`) – `true` to match bins with any descendant, `false` to match leaf bins

compliance: mandatory – This method must be implemented.

`descendant_bin_terms`

`get_bin_query_record`(*bin_record_type*)

Gets the bin query record corresponding to the given `Bin` record `Type`.

Multiple retrievals produce a nested `OR` term.

Parameters **`bin_record_type`** (`osid.type.Type`) – a bin record type

Returns the bin query record

Return type `osid.resource.records.BinQueryRecord`

Raise `NullArgument` – `bin_record_type` is `null`

Raise `OperationFailed` – unable to complete request

Raise `Unsupported` – `has_record_type(bin_record_type)` is `false`

compliance: mandatory – This method must be implemented.

Records

Resource Record

class `dlkit.resource.records.`**`ResourceRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `Resource`.

The methods specified by the record type are available through the underlying object.

Resource Query Record

class `dlkit.resource.records.`**`ResourceQueryRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `ResourceQuery`.

The methods specified by the record type are available through the underlying object.

Resource Form Record

class `dlkit.resource.records.`**`ResourceFormRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `ResourceForm`.

The methods specified by the record type are available through the underlying object.

Resource Search Record

class `dlkit.resource.records.`**`ResourceSearchRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `ResourceSearch`.

The methods specified by the record type are available through the underlying object.

Bin Record

class `dlkit.resource.records.`**`BinRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `Bin`.

The methods specified by the record type are available through the underlying object.

Bin Query Record

class `dlkit.resource.records.`**`BinQueryRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `BinQuery`.

The methods specified by the record type are available through the underlying object.

Bin Form Record

class `dlkit.resource.records.`**`BinFormRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `BinForm`.

The methods specified by the record type are available through the underlying object.

Bin Search Record

class `dlkit.resource.records.`**`BinSearchRecord`**

Bases: *`dlkit.osid.records.OsidRecord`*

A record for a `BinSearch`.

The methods specified by the record type are available through the underlying object.

Type

Summary

Type Open Service Interface Definitions type version 3.0.0

The Type OSID defines a set of interfaces for managing `Type` definitions. `Types` are used as an identifier primarily for identification of interface extensions throughout the OSIDs and occasionally used as an extensible enumeration. An agreement between an OSID Consumer and an OSID Provider means they support the same `Type`.

Types

A `Type` is similar to an Id but includes other data for display and organization. The identification portion of the Type is globally unique and contains:

- authority: the name of the entity or organization responsible for the type. Using a domain name is a reasonable convention.
- identifier: a string serving as an id. The identifier may be a urn, guid, oid or some other means of identification. Since all of the identification elements including the domain and authority create an overall unique Type, the identifier may even be a sequence number defined within a particular domain.
- namespace: a string identifying the namespace of the identifier, such as "urn" or "oid".

Example

Type type = lookupSession.getType("asset", "uri", "http://harvestroad.com/osidTypes/image", "harvestroad.com");

print type.getDisplayName();

The sessions in this OSID offer the capabilities of a `Type` registry to centrally manage definitions and localized display strings and descriptions. Applications may opt to construct their own `Types` directly and bypass this service.

Type Hierarchies

Types are part of an internal hierarchy. A `Type` in a hierarchy includes the `Types` of its children. For example, an `Asset` may have a "photograph" `Type` included as part of an "image" base `Type`.

Unless an application will display a type, it can simply construct a type based on the identification components. OSID Providers may benefit by using this service to manage the type hierarchy, to provide a place to perform mappings across different type definitions, and to provide displayable metadata to its consumers.

Type Type Relations

`Types` may relate to other `Types` to describe constraints or compositions. The relationship is expressed as another Type. For example, a `Position` of type "researcher" may be appropriately associated with an `Organization` of type "laboratory" using a relation Type of "allowed." Or, a root `Event` type depends on a root `TimePeriod` type using a relationship type of "depends on."

Types for Constraints and Side Effects

An OSID Provider may link a `Type,` such as a genus, to a set of constraints that are made known to the application as `Metadata` through an `OsidForm`. Types of an `OsidObject` may also be used by an OSID Provider to constrain the possible relationship `Types` that may be possible to that `OsidObject`. In these uses of `Types,` there is a semantic accompanying the `Type` definition managed within an OSID Provider. The Type OSID manages the metadata of the `Type` itself. Logic implementing the meaning of the `Type` is managed completely within an OSID Provider.

OSIDs emphasize relationships over data typing since type agreements are often an impediment to interoperability. Generally, the rule of thumb for record `Types` is to first explore other `OsidObjects,` even those in other OSIDs for a place for extra data. Often, what is hiding behind a list of data elements is a separate service that can be provided as a separate module and serves to keep the principal `OsidObject` lighter and more flexible.

Genus `Types` primarily serve as a quick and dirty way to unclutter the record `Types` with "is kind of like" tags. `OsidCatalogs` can be used for a richer solution. For example, a genus `Type` may be used to identify all `Events` on a `Calendar` which are classes at a school and is accompanied by constraint logic such that the `Events` occur at a `Location` on campus.

Another pathway to explore is to create a smart `Calendar` from an `EventQuery` that specifies constrraints on the `Event` sponsor, `Location,` or other data required for classes. Creates and updates for Events in that smart `Calendar` will be similarly constrained and surfaced to the OSID Consumer through the `Metadata` in the Event-Forms. While this path is certainly more difficult than simply nailing up some logic indexed by a genus Type, it can be considered if there is a need to expose the logic and authoring capabilities.

OsidPrimitives

Most OSID interfaces are used to encapsulate implementation-specific objects from provider to consumer. `Type` is an `OsidPrimitive` and as such cannot be used to encapsulate implementation-specific data other than what is defined explicitly in the `Type`. An OSID Provider must respect any `Type` constructed by an OSID Consumer. Type Open Service Interface Definitions type version 3.0.0

The Type OSID defines a set of interfaces for managing `Type` definitions. `Types` are used as an identifier primarily for identification of interface extensions throughout the OSIDs and occasionally used as an extensible enumeration. An agreement between an OSID Consumer and an OSID Provider means they support the same `Type`.

Types

A `Type` is similar to an Id but includes other data for display and organization. The identification portion of the Type is globally unique and contains:

- authority: the name of the entity or organization responsible for the type. Using a domain name is a reasonable convention.
- identifier: a string serving as an id. The identifier may be a urn, guid, oid or some other means of identification. Since all of the identification elements including the domain and authority create an overall unique Type, the identifier may even be a sequence number defined within a particular domain.
- namespace: a string identifying the namespace of the identifier, such as "urn" or "oid".

Example

Type type = lookupSession.getType("asset", "uri", "http://harvestroad.com/osidTypes/image", "harvestroad.com");

print type.getDisplayName();

The sessions in this OSID offer the capabilities of a `Type` registry to centrally manage definitions and localized display strings and descriptions. Applications may opt to construct their own `Types` directly and bypass this service.

Type Hierarchies

Types are part of an internal hierarchy. A `Type` in a hierarchy includes the `Types` of its children. For example, an `Asset` may have a "photograph" `Type` included as part of an "image" base `Type`.

Unless an application will display a type, it can simply construct a type based on the identification components. OSID Providers may benefit by using this service to manage the type hierarchy, to provide a place to perform mappings across different type definitions, and to provide displayable metadata to its consumers.

Type Type Relations

`Types` may relate to other `Types` to describe constraints or compositions. The relationship is expressed as another Type. For example, a `Position` of type "researcher" may be appropriately associated with an `Organization` of type "laboratory" using a relation Type of "allowed." Or, a root `Event` type depends on a root `TimePeriod` type using a relationship type of "depends on."

Types for Constraints and Side Effects

An OSID Provider may link a `Type,` such as a genus, to a set of constraints that are made known to the application as `Metadata` through an `OsidForm`. Types of an `OsidObject` may also be used by an OSID Provider to constrain the possible relationship `Types` that may be possible to that `OsidObject`. In these uses of `Types,` there is a semantic accompanying the `Type` definition managed within an OSID Provider. The Type OSID manages the metadata of the `Type` itself. Logic implementing the meaning of the `Type` is managed completely within an OSID Provider.

OSIDs emphasize relationships over data typing since type agreements are often an impediment to interoperability. Generally, the rule of thumb for record `Types` is to first explore other `OsidObjects,` even those in other OSIDs for a place for extra data. Often, what is hiding behind a list of data elements is a separate service that can be provided as a separate module and serves to keep the principal `OsidObject` lighter and more flexible.

Genus `Types` primarily serve as a quick and dirty way to unclutter the record `Types` with "is kind of like" tags. `OsidCatalogs` can be used for a richer solution. For example, a genus `Type` may be used to identify all `Events` on a `Calendar` which are classes at a school and is accompanied by constraint logic such that the `Events` occur at a `Location` on campus.

Another pathway to explore is to create a smart `Calendar` from an `EventQuery` that specifies constrraints on the `Event` sponsor, `Location,` or other data required for classes. Creates and updates for Events in that smart `Calendar` will be similarly constrained and surfaced to the OSID Consumer through the `Metadata` in the Event-Forms. While this path is certainly more difficult than simply nailing up some logic indexed by a genus Type, it can be considered if there is a need to expose the logic and authoring capabilities.

OsidPrimitives

Most OSID interfaces are used to encapsulate implementation-specific objects from provider to consumer. `Type` is an `OsidPrimitive` and as such cannot be used to encapsulate implementation-specific data other than what is defined explicitly in the `Type`. An OSID Provider must respect any `Type` constructed by an OSID Consumer.

Service Managers

Type Profile

class `dlkit.services.type.`**`TypeProfile`**

Bases: `dlkit.osid.managers.OsidProfile`

The `TypeProfile` describes the interoperability among type services.

Type Manager

class `dlkit.services.type.`**`TypeManager`**(*proxy=None*)

Bases: `dlkit.osid.managers.OsidManager`, `dlkit.osid.sessions.OsidSession`, *`dlkit.services.type.TypeProfile`*

This manager provides access to the available sessions of the type service.

The `TypeLookupSession` is used for looking up `Types` and the `TypeAdminSession` is used for managing and registering new Types.

Type Methods

Type Methods

Type Proxy Methods

Methods

`TypeManager.`**`display_name`**

`TypeManager.`**`display_label`**

`TypeManager.`**`description`**

`TypeManager.`**`domain`**

`TypeManager.`**`authority`**

Gets the authority of this `Type`.

The authority is a string used to ensure the uniqueness of this `Type` when using a non- federated identifier space. Generally, it is a domain name identifying the party responsible for this `Type`. This method is used to compare one `Type` to another.

Returns the authority of this `Type`

Return type `string`

compliance: mandatory – This method must be implemented.

`TypeManager.`**`namespace`**

Gets the namespace of the identifier.

This method is used to compare one `Type` to another.

Returns the authority of this `Type`

Return type `string`

compliance: mandatory – This method must be implemented.

`TypeManager.`**`identifier`**

Gets the identifier of this `Type`.

This method is used to compare one `Type` to another.

Returns the identifier of this `Type`

Return type `string`

compliance: mandatory – This method must be implemented.

Type Methods

`TypeManager.`**`display_name_metadata`**

Gets the metadata for the display name.

Returns metadata for the display name

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`TypeManager.`**`display_name`**

`TypeManager.`**`display_label_metadata`**

Gets the metadata for the display label.

Returns metadata for the display label

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`TypeManager.`**`display_label`**

`TypeManager.`**`description_metadata`**

Gets the metadata for the description.

Returns metadata for the description

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`TypeManager.`**`description`**

`TypeManager.`**`domain_metadata`**

Gets the metadata for the domain.

Returns metadata for the domain

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

`TypeManager.`**`domain`**

Type Methods

`TypeManager.`**`next_type`**

Gets the next `Type` in this list.

Returns the next `Type` in this list. The `has_next()` method should be used to test that a next `Type` is available before calling this method.

Return type `osid.type.Type`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

`TypeManager.`**`get_next_types`**`(`*n*`)`

Gets the next set of `Types` in this list.

The specified amount must be less than or equal to the return from `available()`.

Parameters **n** (`cardinal`) – the number of `Type` elements requested which must be less than or equal to `available()`

Returns an array of `Type` elements.The length of the array is less than or equal to the number specified.

Return type `osid.type.Type`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

Primitives

Type

class `dlkit.type.primitives.`**`Type`**

Bases: *`dlkit.osid.markers.OsidPrimitive`*

The Type is a form of identifier that is primarily used to identify interface specifications.

The `Type` differs from `Id` in that it offers display information and `Types` may be arranged in hierarchies to indicate an extended interface. Semantically, an `Id` identifies any OSID object while the `Type` identifies a specification.

The components of the Type that make up its identification are:

- identifier: a unique key or guid
- namespace: the namespace of the identifier
- authority: the isuer of the identifier

Persisting a type reference means to persist the above identification elements. In addition to these identifier components, A `Type` mai also provide some additional metadata such as a name, description and domain.

`display_name`

Gets the full display name of this `Type`.

Returns the display name of this `Type`

Return type `osid.locale.DisplayText`

compliance: mandatory – This method must be implemented.

`display_label`

Gets the shorter display label for this `Type`.

Where a display name of a `Type` might be `"` Critical Logging Priority Type", the display label could be "critical".

Returns the display label for this `Type`

Return type `osid.locale.DisplayText`

compliance: mandatory – This method must be implemented.

`description`

Gets a description of this `Type`.

Returns the description of this `Type`

Return type `osid.locale.DisplayText`

compliance: mandatory – This method must be implemented.

`domain`

Gets the domain.

The domain can provide an information label about ths application space of this Type.

Returns the domain of this `Type`

Return type `osid.locale.DisplayText`

compliance: mandatory – This method must be implemented.

authority

Gets the authority of this `Type`.

The authority is a string used to ensure the uniqueness of this `Type` when using a non- federated identifier space. Generally, it is a domain name identifying the party responsible for this `Type`. This method is used to compare one `Type` to another.

Returns the authority of this `Type`

Return type `string`

compliance: mandatory – This method must be implemented.

namespace

Gets the namespace of the identifier.

This method is used to compare one `Type` to another.

Returns the authority of this `Type`

Return type `string`

compliance: mandatory – This method must be implemented.

identifier

Gets the identifier of this `Type`.

This method is used to compare one `Type` to another.

Returns the identifier of this `Type`

Return type `string`

compliance: mandatory – This method must be implemented.

Objects

Type Form

class `dlkit.type.objects.`**TypeForm**

Bases: *dlkit.osid.objects.OsidForm*

This form provides a means of updating various fields in the `Type`.

Note that the domain, authority and identifier are part of the `Type` identification, and as such not modifiable.

display_name_metadata

Gets the metadata for the display name.

Returns metadata for the display name

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

display_name

display_label_metadata

Gets the metadata for the display label.

Returns metadata for the display label

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

display_label

description_metadata
Gets the metadata for the description.

Returns metadata for the description

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

description

domain_metadata
Gets the metadata for the domain.

Returns metadata for the domain

Return type `osid.Metadata`

compliance: mandatory – This method must be implemented.

domain

Type List

class `dlkit.type.objects.`**TypeList**
Bases: *`dlkit.osid.objects.OsidList`*

Like all `OsidLists`, `TypeList` provides a means for accessing `Type` elements sequentially either one at a time or many at a time.

Examples: while (tl.hasNext()) { Type type = tl.getNextType(); }

or

while (tl.hasNext()) { Type[] types = tl.getNextTypes(tl.available());

}

next_type
Gets the next `Type` in this list.

Returns the next `Type` in this list. The `has_next()` method should be used to test that a next `Type` is available before calling this method.

Return type `osid.type.Type`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

get_next_types(*n*)
Gets the next set of `Types` in this list.

The specified amount must be less than or equal to the return from `available()`.

Parameters **n** (`cardinal`) – the number of `Type` elements requested which must be less than or equal to `available()`

Returns an array of `Type` elements.The length of the array is less than or equal to the number specified.

Return type `osid.type.Type`

Raise `IllegalState` – no more elements available in this list

Raise `OperationFailed` – unable to complete request

compliance: mandatory – This method must be implemented.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index