dlkit Documentation

Release 0.0.2

Jeff Merriman

Sep 27, 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 160 service packages and 9,000 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.

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

CHAPTER 1

Contents:

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`
`TypeLookupSession.get_type`

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 Manager

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

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`

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`

Assessment Profile Methods

`AssessmentManager.`**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`

`AssessmentManager.`**`supports_item_lookup`**`()`
Tests if an item lookup service is supported.

Returns true if item lookup is supported, false otherwise

Return type `boolean`

`AssessmentManager.`**`supports_item_query`**`()`
Tests if an item query service is supported.

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

Return type `boolean`

`AssessmentManager.`**`supports_item_admin`**`()`
Tests if an item administrative service is supported.

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

Return type `boolean`

`AssessmentManager.`**`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`

`AssessmentManager.`**`supports_assessment_query`**`()`
Tests if an assessment query service is supported.

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

Return type `boolean`

`AssessmentManager.`**`supports_assessment_admin`**`()`
Tests if an assessment administrative service is supported.

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

Return type `boolean`

`AssessmentManager.`**`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`

`AssessmentManager.`**`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`

`AssessmentManager.`**`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`

AssessmentManager.**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`

AssessmentManager.**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`

AssessmentManager.**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`

AssessmentManager.**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`

AssessmentManager.**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`

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

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

Return type `boolean`

AssessmentManager.**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`

AssessmentManager.**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`

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

Returns a list containing the supported `Item` record types

Return type `osid.type.TypeList`

AssessmentManager.**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`

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

Returns a list containing the supported `Assessment` record types

Return type `osid.type.TypeList`

`AssessmentManager.`**`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`

`AssessmentManager.`**`assessment_offered_record_types`**
Gets the supported `AssessmentOffered` record types.

Returns a list containing the supported `AssessmentOffered` record types

Return type `osid.type.TypeList`

`AssessmentManager.`**`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`

`AssessmentManager.`**`assessment_taken_record_types`**
Gets the supported `AssessmentTaken` record types.

Returns a list containing the supported `AssessmentTaken` record types

Return type `osid.type.TypeList`

`AssessmentManager.`**`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`

`AssessmentManager.`**`assessment_section_record_types`**
Gets the supported `AssessmentSection` record types.

Returns a list containing the supported `AssessmentSection` record types

Return type `osid.type.TypeList`

`AssessmentManager.`**`bank_record_types`**
Gets the supported `Bank` record types.

Returns a list containing the supported `Bank` record types

Return type `osid.type.TypeList`

`AssessmentManager.`**`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`

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

AssessmentManager.**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.

AssessmentManager.**use_plenary_bank_view**()
A complete view of the 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.

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

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

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

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

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

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

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`

`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`

`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

`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()`

`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

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

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()

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

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

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

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

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

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

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

AssessmentManager.**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.

AssessmentManager.**use_plenary_bank_view**()
A complete view of the 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Bank

Bank

class dlkit.services.assessment.**Bank**
Bases: dlkit.osid.objects.OsidCatalog, dlkit.osid.sessions.OsidSession

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

Assessment Methods

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`

`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

`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

`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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

`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

`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

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

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

Bank.**get_responses**(*assessment_section_id*)
Gets all submitted responses.

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

Returns the list of responses

Return type osid.assessment.ResponseList

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

`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

`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

`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

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

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

Item Lookup Methods

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

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.

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.

Bank.**use_federated_bank_view**()
Federates the view for methods in this session. A federated view will include assessment items in assessment banks which are children of this assessment bank in the assessment bank hierarchy.

Bank.**use_isolated_bank_view**()
Isolates the view for methods in this session. An isolated view restricts lookups to this assessment bank only.

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

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

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

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

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

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

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

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

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

Item Query Methods

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

Bank.**use_federated_bank_view**()
Federates the view for methods in this session. A federated view will include assessment items in assessment banks which are children of this assessment bank in the assessment bank hierarchy.

Bank.**use_isolated_bank_view**()
Isolates the view for methods in this session. An isolated view restricts lookups to this assessment bank only.

Bank.**item_query**
Gets an assessment item query.

Returns the assessment item query

Return type osid.assessment.ItemQuery

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

Item Admin Methods

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

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

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

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()

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

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

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()

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

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

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

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

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

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

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

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()

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

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

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()

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

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

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

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

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

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()

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

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

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()

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

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

Assessment Lookup Methods

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

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.

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.

Bank.**use_federated_bank_view**()
Federates the view for methods in this session. A federated view will include assessment items in assessment banks which are children of this assessment bank in the assessment bank hierarchy.

Bank.**use_isolated_bank_view**()
Isolates the view for methods in this session. An isolated view restricts lookups to this assessment bank only.

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

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

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

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

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

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

Assessment Query Methods

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

Bank.**use_federated_bank_view**()
Federates the view for methods in this session. A federated view will include assessment items in assessment banks which are children of this assessment bank in the assessment bank hierarchy.

Bank.**use_isolated_bank_view**()
Isolates the view for methods in this session. An isolated view restricts lookups to this assessment bank only.

Bank.**assessment_query**
Gets an assessment query.

Returns the assessment query

Return type osid.assessment.AssessmentQuery

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

Assessment Admin Methods

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

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

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

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()

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

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

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()

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

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

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

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

Assessment Basic Authoring Methods

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

Bank.**get_items**(*assessment_taken_id*)

Gets the items questioned in a assessment.

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

Returns the list of assessment questions

Return type osid.assessment.ItemList

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

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

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

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

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

Assessment Offered Lookup Methods

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

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.

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.

Bank.**use_federated_bank_view**()
Federates the view for methods in this session. A federated view will include assessment items in assessment banks which are children of this assessment bank in the assessment bank hierarchy.

Bank.**use_isolated_bank_view**()
Isolates the view for methods in this session. An isolated view restricts lookups to this assessment bank only.

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

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

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

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

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

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

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

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

Assessment Offered Query Methods

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

Bank.**use_federated_bank_view**()
Federates the view for methods in this session. A federated view will include assessment items in assessment banks which are children of this assessment bank in the assessment bank hierarchy.

Bank.**use_isolated_bank_view**()
Isolates the view for methods in this session. An isolated view restricts lookups to this assessment bank only.

Bank.**assessment_offered_query**
Gets an assessment offered query.

Returns the assessment offered query

Return type osid.assessment.AssessmentOfferedQuery

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

Assessment Offered Admin Methods

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

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

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

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()

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

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

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()

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

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

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

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

Assessment Taken Lookup Methods

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

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.

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.

Bank.**use_federated_bank_view**()
Federates the view for methods in this session. A federated view will include assessment items in assessment banks which are children of this assessment bank in the assessment bank hierarchy.

Bank.**use_isolated_bank_view**()
Isolates the view for methods in this session. An isolated view restricts lookups to this assessment bank only.

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

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

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

`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

`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

`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

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

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

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

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

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

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

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

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

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

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

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

Assessment Taken Query Methods

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`

`Bank.`**`use_federated_bank_view`**`()`
Federates the view for methods in this session. A federated view will include assessment items in assessment banks which are children of this assessment bank in the assessment bank hierarchy.

`Bank.`**`use_isolated_bank_view`**`()`
Isolates the view for methods in this session. An isolated view restricts lookups to this assessment bank only.

`Bank.`**`assessment_taken_query`**
Gets an assessment taken query.

Returns the assessment taken query

Return type `osid.assessment.AssessmentTakenQuery`

`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

Assessment Taken Admin Methods

`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`

`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

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

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()

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

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

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()

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

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

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

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

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

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

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

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

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

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

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

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

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`

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

question_id
Gets the `Id` of the `Question`.

Returns the question `Id`

Return type `osid.id.Id`

question
Gets the question.

Returns the question

Return type osid.assessment.Question

Raise OperationFailed – unable to complete request

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

answers
Gets the answers.

Returns the answers

Return type osid.assessment.AnswerList

Raise OperationFailed – unable to complete request

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

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

learning_objectives
Sets the learning objectives.

Parameters **objective_ids** (osid.id.Id[]) – the learning objective Ids

Raise InvalidArgument – objective_ids is invalid

Raise NoAccess – Metadata.isReadOnly() is true

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

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

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

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`

level
Gets the `Grade` corresponding to the assessment difficulty.

Returns the level

Return type `osid.grading.Grade`

Raise `OperationFailed` – unable to complete request

has_rubric()
Tests if a rubric assessment is associated with this assessment.

Returns `true` if a rubric is available, `false` otherwise

Return type `boolean`

rubric_id
Gets the `Id` of the rubric.

Returns an assessment `Id`

Return type `osid.id.Id`

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

rubric
Gets the rubric.

Returns the assessment

Return type `osid.assessment.Assessment`

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

Raise `OperationFailed` – unable to complete request

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

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

level
Sets the level of difficulty expressed as a Grade.

Parameters **grade_id** (osid.id.Id) – the grade level

Raise InvalidArgument – grade_id is invalid

Raise NoAccess – Metadata.isReadOnly() is true

Raise NullArgument – grade_id is null

rubric_metadata
Gets the metadata for a rubric assessment.

Returns metadata for the assesment

Return type osid.Metadata

rubric
Sets the rubric expressed as another assessment.

Parameters **assessment_id** (osid.id.Id) – the assessment Id

Raise InvalidArgument – assessment_id is invalid

Raise NoAccess – Metadata.isReadOnly() is true

Raise NullArgument – assessment_id is null

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`

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

`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

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`

assessment
Gets the assessment corresponding to this assessment offereng.

Returns the assessment

Return type `osid.assessment.Assessment`

Raise `OperationFailed` – unable to complete request

level_id
Gets the `Id` of a `Grade` corresponding to the assessment difficulty.

Returns a grade id

Return type `osid.id.Id`

level
Gets the `Grade` corresponding to the assessment difficulty.

Returns the level

Return type `osid.grading.Grade`

Raise `OperationFailed` – unable to complete request

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`

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`

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`

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`

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

deadline
Gets the end time for this assessment.

Returns the designated end time

Return type osid.calendaring.DateTime

Raise IllegalState – has_deadline() is false

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

duration
Gets the duration for this assessment.

Returns the duration

Return type osid.calendaring.Duration

Raise IllegalState – has_duration() is false

is_scored()
Tests if this assessment will be scored.

Returns true if this assessment will be scored false otherwise

Return type boolean

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

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

is_graded()
Tests if this assessment will be graded.

Returns true if this assessment will be graded, false otherwise

Return type boolean

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

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

has_rubric()
Tests if a rubric assessment is associated with this assessment.

Returns true if a rubric is available, false otherwise

Return type boolean

rubric_id
Gets the Id of the rubric.

Returns an assessment offered Id

Return type osid.id.Id

Raise IllegalState – has_rubric() is false

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

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

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`

`level`
Sets the level of difficulty expressed as a `Grade`.

Parameters **`grade_id`** (`osid.id.Id`) – the grade level

Raise `InvalidArgument` – `grade_id` is invalid

Raise `NoAccess` – `Metadata.isReadOnly()` is `true`

`items_sequential_metadata`
Gets the metadata for sequential operation.

Returns metadata for the sequential flag

Return type `osid.Metadata`

`items_sequential`
Sets the items sequential flag.

Parameters **`sequential`** (`boolean`) – `true` if the items are taken sequentially, `false` if the items can be skipped and revisited

Raise `InvalidArgument` – `sequential` is invalid

Raise `NoAccess` – `Metadata.isReadOnly()` is `true`

`items_shuffled_metadata`
Gets the metadata for shuffling items.

Returns metadata for the shuffled flag

Return type `osid.Metadata`

`items_shuffled`
Sets the shuffle flag.

The shuffle flag may be overidden by other assessment sequencing rules.

Parameters **`shuffle`** (`boolean`) – `true` if the items are shuffled, `false` if the items appear in the designated order

Raise `InvalidArgument` – `shuffle` is invalid

Raise `NoAccess` – `Metadata.isReadOnly()` is `true`

`start_time_metadata`
Gets the metadata for the assessment start time.

Returns metadata for the start time

Return type `osid.Metadata`

`start_time`
Sets the assessment start time.

Parameters **start** (osid.calendaring.DateTime) – assessment start time

Raise InvalidArgument – start is invalid

Raise NoAccess – Metadata.isReadOnly() is true

deadline_metadata
Gets the metadata for the assessment deadline.

Returns metadata for the end time

Return type osid.Metadata

deadline
Sets the assessment end time.

Parameters **end** (timestamp) – assessment end time

Raise InvalidArgument – end is invalid

Raise NoAccess – Metadata.isReadOnly() is true

duration_metadata
Gets the metadata for the assessment duration.

Returns metadata for the duration

Return type osid.Metadata

duration
Sets the assessment duration.

Parameters **duration** (osid.calendaring.Duration) – assessment duration

Raise InvalidArgument – duration is invalid

Raise NoAccess – Metadata.isReadOnly() is true

score_system_metadata
Gets the metadata for a score system.

Returns metadata for the grade system

Return type osid.Metadata

score_system
Sets the scoring system.

Parameters **grade_system_id** (osid.id.Id) – the grade system

Raise InvalidArgument – grade_system_id is invalid

Raise NoAccess – Metadata.isReadOnly() is true

grade_system_metadata
Gets the metadata for a grading system.

Returns metadata for the grade system

Return type osid.Metadata

grade_system
Sets the grading system.

Parameters **grade_system_id** (osid.id.Id) – the grade system

Raise InvalidArgument – grade_system_id is invalid

Raise NoAccess – Metadata.isReadOnly() is true

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

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

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

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

assessment_offered
Gets the AssessmentOffered.

Returns the assessment offered

Return type osid.assessment.AssessmentOffered

Raise OperationFailed – unable to complete request

taker_id
Gets the Id of the resource who took or is taking this assessment.

Returns the resource Id

Return type osid.id.Id

taker
Gets the Resource taking this assessment.

Returns the resource

Return type osid.resource.Resource

Raise OperationFailed – unable to complete request

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

taking_agent
Gets the Agent.

Returns the agent

Return type osid.authentication.Agent

Raise OperationFailed – unable to complete request

has_started()
Tests if this assessment has begun.

Returns true if the assessment has begun, false otherwise

Return type boolean

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`

has_ended()
Tests if this assessment has ended.

Returns `true` if the assessment has ended, `false` otherwise

Return type `boolean`

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`

time_spent
Gets the total time spent taking this assessment.

Returns the total time spent

Return type `osid.calendaring.Duration`

completion
Gets a completion percentage of the assessment.

Returns the percent complete (0-100)

Return type `cardinal`

is_scored()
Tests if a score is available for this assessment.

Returns `true` if a score is available, `false` otherwise

Return type `boolean`

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`

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

score
Gets a score for the assessment.

Returns the score

Return type `decimal`

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

is_graded()
Tests if a grade is available for this assessment.

Returns `true` if a grade is available, `false` otherwise

Return type `boolean`

grade_id
Gets a grade `Id` for the assessment.

Returns the grade

Return type `osid.id.Id`

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

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

feedback
Gets any overall comments available for this assessment by the grader.

Returns comments

Return type `osid.locale.DisplayText`

has_rubric()
Tests if a rubric assessment is associated with this assessment.

Returns `true` if a rubric is available, `false` otherwise

Return type `boolean`

rubric_id
Gets the `Id` of the rubric.

Returns an assessment taken `Id`

Return type `osid.id.Id`

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

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

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

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

taker
Sets the resource who will be taking this assessment.

Parameters **resource_id** (osid.id.Id) – the resource Id

Raise InvalidArgument – resource_id is invalid

Raise NoAccess – Metadata.isReadOnly() is true

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

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

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

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

assessment_taken
Gets the AssessmentTakeb.

Returns the assessment taken

Return type osid.assessment.AssessmentTaken

Raise OperationFailed – unable to complete request

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`

allocated_time
Gets the allocated time for this section.

Returns allocated time

Return type `osid.calendaring.Duration`

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

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`

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`

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`

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

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

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`

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

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

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

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

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`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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`

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`

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`

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`

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

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`

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`

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`

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

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`

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`

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`

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

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

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

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

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

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

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

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`

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

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`

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`

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`

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`

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

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

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

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

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

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`

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

level_terms

match_items_sequential(*match*)
Match sequential assessments.

Parameters **match** (`boolean`) – `true` for a positive match, `false` for a negative match

items_sequential_terms

match_items_shuffled(*match*)
Match shuffled item assessments.

Parameters **match** (`boolean`) – `true` for a positive match, `false` for a negative match

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`

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

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`

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

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

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

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

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

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

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

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`

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`

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`

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

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`

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`

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

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

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

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

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

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

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

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

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`

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`

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`

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`

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`

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`

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`

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`

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`

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`

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`

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`

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

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`

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

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`

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`

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`

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`

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

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`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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`

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`

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`

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`

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

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`

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`

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`

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

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`

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`

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`

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

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

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

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

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

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

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

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`

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

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`

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.

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 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 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 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.

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

item
Gets the Item.

Returns the assessment item

Return type osid.assessment.Item

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

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 Manager

class dlkit.services.commenting.**CommentingManager**
Bases: dlkit.osid.managers.OsidManager, dlkit.osid.sessions.OsidSession, dlkit.services.commenting.CommentingProfile

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

Commenting Profile Methods

CommentingManager.**supports_comment_lookup**()
Tests for the availability of a comment lookup service.

Returns true if comment lookup is available, false otherwise

Return type boolean

CommentingManager.**supports_comment_query**()
Tests if querying comments is available.

Returns true if comment query is available, false otherwise

Return type boolean

CommentingManager.**supports_comment_admin**()
Tests if managing comments is available.

Returns true if comment admin is available, false otherwise

Return type boolean

CommentingManager.**supports_book_lookup**()
Tests for the availability of an book lookup service.

Returns true if book lookup is available, false otherwise

Return type boolean

CommentingManager.**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`

CommentingManager.**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`

CommentingManager.**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`

CommentingManager.**comment_record_types**
Gets the supported `Comment` record types.

Returns a list containing the supported comment record types

Return type `osid.type.TypeList`

CommentingManager.**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`

CommentingManager.**book_record_types**
Gets the supported `Book` record types.

Returns a list containing the supported book record types

Return type `osid.type.TypeList`

CommentingManager.**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`

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`

CommentingManager.**use_comparative_book_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.

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.

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

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

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

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

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

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

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

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

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

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()

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

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

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()

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

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

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

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

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

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

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

CommentingManager.**use_comparative_book_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.

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.

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

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

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

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

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

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

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 true 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

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

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

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

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

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

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

`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

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`

`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

`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

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

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

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

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

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

Book

Book

class dlkit.services.commenting.**Book**
Bases: dlkit.osid.objects.OsidCatalog, dlkit.osid.sessions.OsidSession

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

Comment Lookup Methods

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

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.

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.

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.

Book.**use_isolated_book_view**()
Isolates the view for methods in this session. An isolated view restricts retrievals to this book only.

Book.**use_effective_comment_view**()
Only comments whose effective dates are current are returned by methods in this session.

Book.**use_any_effective_comment_view**()
All comments of any effective dates are returned by all methods in this session.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- **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 – reference_id, comment_genus_type, from, or to is null

Raise OperationFailed – unable to complete request

Raise PermissionDenied – authorization failure

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

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

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

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

`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

Comment Query Methods

`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`

`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.

`Book.`**`use_isolated_book_view`**`()`
Isolates the view for methods in this session. An isolated view restricts retrievals to this book only.

`Book.`**`comment_query`**
Gets a comment query.

Returns the comment query

Return type `osid.commenting.CommentQuery`

`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

Comment Admin Methods

`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

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

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

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()

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

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

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()

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

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

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

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

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

commentor_id
Gets the Id of the resource who created this comment.

Returns the Resource Id

Return type osid.id.Id

commentor
Gets the resource who created this comment.

Returns the Resource

Return type osid.resource.Resource

Raise OperationFailed – unable to complete request

commenting_agent_id
Gets the Id of the agent who created this comment.

Returns the Agent Id

Return type osid.id.Id

commenting_agent
Gets the agent who created this comment.

Returns the Agent

Return type osid.authentication.Agent

Raise OperationFailed – unable to complete request

text
Gets the comment text.

Returns the comment text

Return type osid.locale.DisplayText

has_rating()
Tests if this comment includes a rating.

Returns true if this comment includes a rating, false otherwise

Return type boolean

rating_id
Gets the Id of the Grade.

Returns the Agent Id

Return type osid.id.Id

Raise IllegalState – has_rating() is false

rating
Gets the Grade.

Returns the Grade

Return type osid.grading.Grade

Raise IllegalState – has_rating() is false

Raise OperationFailed – unable to complete request

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

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

text
Sets the text.

Parameters **text** (string) – the new text

Raise InvalidArgument – text is invalid

Raise NoAccess – Metadata.isReadOnly() is true

Raise NullArgument – text is null

rating_metadata
Gets the metadata for a rating.

Returns metadata for the rating

Return type osid.Metadata

rating
Sets the rating.

Parameters **grade_id** (osid.id.Id) – the new rating

Raise InvalidArgument – grade_id is invalid

Raise NoAccess – Metadata.isReadOnly() is true

Raise NullArgument – grade_id is null

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

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

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

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

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

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

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

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

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

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

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

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

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`

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`

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

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`

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`

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`

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

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

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

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

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

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`

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`

comment_query
Gets the query for a comment.

Returns the comment query

Return type `osid.commenting.CommentQuery`

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

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

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`

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`

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`

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

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

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

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

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

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

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.

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.

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 Manager

class dlkit.services.learning.**LearningManager**

Bases: dlkit.osid.managers.OsidManager, dlkit.osid.sessions.OsidSession, dlkit.services.learning.LearningProfile

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

Learning Profile Methods

LearningManager.**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

LearningManager.**supports_objective_admin**()
Tests if an objective administrative service is supported.

Returns true if objective admin is supported, false otherwise

Return type boolean

LearningManager.**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

LearningManager.**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

LearningManager.**supports_objective_sequencing**()
Tests if an objective sequencing design is supported.

Returns true if objective sequencing is supported, false otherwise

Return type boolean

LearningManager.**supports_objective_requisite**()
Tests if an objective requisite service is supported.

Returns true if objective requisite service is supported, false otherwise

Return type boolean

LearningManager.**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

LearningManager.**supports_activity_lookup**()
Tests if an activity lookup service is supported.

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

Return type `boolean`

`LearningManager.`**`supports_activity_admin`**`()`
Tests if an activity administrative service is supported.

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

Return type `boolean`

`LearningManager.`**`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`

`LearningManager.`**`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`

`LearningManager.`**`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`

`LearningManager.`**`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`

`LearningManager.`**`objective_record_types`**
Gets the supported `Objective` record types.

Returns a list containing the supported `Objective` record types

Return type `osid.type.TypeList`

`LearningManager.`**`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`

`LearningManager.`**`activity_record_types`**
Gets the supported `Activity` record types.

Returns a list containing the supported `Activity` record types

Return type `osid.type.TypeList`

`LearningManager.`**`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`

LearningManager.**proficiency_record_types**
Gets the supported Proficiency record types.

Returns a list containing the supported Proficiency record types

Return type osid.type.TypeList

LearningManager.**proficiency_search_record_types**
Gets the supported Proficiency search types.

Returns a list containing the supported Proficiency search types

Return type osid.type.TypeList

LearningManager.**objective_bank_record_types**
Gets the supported ObjectiveBank record types.

Returns a list containing the supported ObjectiveBank record types

Return type osid.type.TypeList

LearningManager.**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

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

LearningManager.**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.

LearningManager.**use_plenary_objective_bank_view**()
A complete view of the 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.

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

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

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

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

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

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

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

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

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.

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()

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

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

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()

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

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

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

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

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

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

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

LearningManager.**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.

LearningManager.**use_plenary_objective_bank_view**()
A complete view of the 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.

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

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

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

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

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

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

`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

`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

`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

`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

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

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

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

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

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

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

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

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

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

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

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

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

Objective Bank

Objective Bank

class dlkit.services.learning.**ObjectiveBank**
Bases: dlkit.osid.objects.OsidCatalog, dlkit.osid.sessions.OsidSession

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

Objective Lookup Methods

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

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.

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.

ObjectiveBank.**use_federated_objective_bank_view**()
Federates the view for methods in this session. A federated view will include objectives in objective banks which are children of this objective bank in the objective bank hierarchy.

ObjectiveBank.**use_isolated_objective_bank_view**()
Isolates the view for methods in this session. An isolated view restricts lookups to this objective bank only.

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

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

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

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

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

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

Objective Admin Methods

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

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

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

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()

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

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

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()

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

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

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`

`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

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`

`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

`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`

`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.

`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.

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

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

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

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

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

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

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

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

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

`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

`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

`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

`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

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

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

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

`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`

`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

`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

`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

`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

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

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

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

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

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

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

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

Objective Requisite Methods

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`

`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.

`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.

`ObjectiveBank.`**`use_federated_objective_bank_view`**`()`
Federates the view for methods in this session. A federated view will include objectives in objective banks which are children of this objective bank in the objective bank hierarchy.

`ObjectiveBank.`**`use_isolated_objective_bank_view`**`()`
Isolates the view for methods in this session. An isolated view restricts lookups to this objective bank only.

`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

`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

`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

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

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

Objective Requisite Assignment Methods

`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`

`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

`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

`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

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

Activity Lookup Methods

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

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.

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.

ObjectiveBank.**use_federated_objective_bank_view**()

Federates the view for methods in this session. A federated view will include objectives in objective banks which are children of this objective bank in the objective bank hierarchy.

ObjectiveBank.**use_isolated_objective_bank_view**()

Isolates the view for methods in this session. An isolated view restricts lookups to this objective bank only.

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

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

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

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

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

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

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

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

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

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

Activity Admin Methods

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

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

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

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()

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

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

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()

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

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

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

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

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

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

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

has_knowledge_category()
Tests if this objective has a knowledge dimension.

Returns `true` if a knowledge category exists, `false` otherwise

Return type `boolean`

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`

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

has_cognitive_process()
Tests if this objective has a cognitive process type.

Returns `true` if a cognitive process exists, `false` otherwise

Return type `boolean`

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`

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

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

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

assessment
Sets the assessment.

Parameters **assessment_id** (osid.id.Id) – the new assessment

Raise InvalidArgument – assessment_id is invalid

Raise NoAccess – assessment_id cannot be modified

Raise NullArgument – assessment_id is null

knowledge_category_metadata
Gets the metadata for a knowledge category.

Returns metadata for the knowledge category

Return type osid.Metadata

knowledge_category
Sets the knowledge category.

Parameters **grade_id** (osid.id.Id) – the new knowledge category

Raise InvalidArgument – grade_id is invalid

Raise NoAccess – grade_id cannot be modified

Raise NullArgument – grade_id is null

cognitive_process_metadata
Gets the metadata for a cognitive process.

Returns metadata for the cognitive process

Return type osid.Metadata

cognitive_process
Sets the cognitive process.

Parameters **grade_id** (osid.id.Id) – the new cognitive process

Raise InvalidArgument – grade_id is invalid

Raise NoAccess – grade_id cannot be modified

Raise NullArgument – grade_id is null

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

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

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

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

objective
Gets the related objective.

Returns the related objective

Return type osid.learning.Objective

Raise OperationFailed – unable to complete request

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

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

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

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

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`

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

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`

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`

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

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`

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

assets
Sets the assets.

Parameters **asset_ids** (osid.id.Id[]) – the asset Ids

Raise InvalidArgument – asset_ids is invalid

Raise NullArgument – asset_ids is null

Raise NoAccess – Metadata.isReadOnly() is true

courses_metadata
Gets the metadata for the courses.

Returns metadata for the courses

Return type osid.Metadata

courses
Sets the courses.

Parameters **course_ids** (osid.id.Id[]) – the course Ids

Raise InvalidArgument – course_ids is invalid

Raise NullArgument – course_ids is null

Raise NoAccess – Metadata.isReadOnly() is true

assessments_metadata
Gets the metadata for the assessments.

Returns metadata for the assessments

Return type osid.Metadata

assessments
Sets the assessments.

Parameters **assessment_ids** (osid.id.Id[]) – the assessment Ids

Raise InvalidArgument – assessment_ids is invalid

Raise NullArgument – assessment_ids is null

Raise NoAccess – Metadata.isReadOnly() is true

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

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

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

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`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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`

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`

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`

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

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`

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`

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`

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

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

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

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

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

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

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

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`

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

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`

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`

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`

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

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`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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`

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`

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`

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

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`

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`

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`

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

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`

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`

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`

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

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`

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.

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.

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.

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 Manager

class dlkit.services.repository.**RepositoryManager**
Bases: dlkit.osid.managers.OsidManager, dlkit.osid.sessions.OsidSession, dlkit.services.repository.RepositoryProfile

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

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

Repository Profile Methods

RepositoryManager.**supports_asset_lookup**()
Tests if asset lookup is supported.

Returns true if asset lookup is supported , false otherwise

Return type boolean

RepositoryManager.**supports_asset_query**()
Tests if asset query is supported.

Returns true if asset query is supported , false otherwise

Return type boolean

RepositoryManager.**supports_asset_admin**()
Tests if asset administration is supported.

Returns true if asset administration is supported, false otherwise

Return type boolean

RepositoryManager.**supports_repository_lookup**()
Tests if repository lookup is supported.

Returns true if repository lookup is supported , false otherwise

Return type boolean

RepositoryManager.**supports_repository_admin**()
Tests if repository administration is supported.

Returns true if repository administration is supported, false otherwise

Return type boolean

RepositoryManager.**asset_record_types**
Gets all the asset record types supported.

Returns the list of supported asset record types

Return type osid.type.TypeList

RepositoryManager.**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

RepositoryManager.**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

RepositoryManager.**composition_record_types**
Gets all the composition record types supported.

Returns the list of supported composition record types

Return type osid.type.TypeList

RepositoryManager.**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

RepositoryManager.**repository_record_types**
Gets all the repository record types supported.

Returns the list of supported repository record types

Return type `osid.type.TypeList`

`RepositoryManager.`**`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`

`RepositoryManager.`**`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`

`RepositoryManager.`**`coordinate_types`**
Gets all the coordinate types supported.

Returns the list of supported coordinate types

Return type `osid.type.TypeList`

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`

`RepositoryManager.`**`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.

`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.

`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

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

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

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

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

`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

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`

`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`

`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

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()

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

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

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()

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

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

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

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

Repository

Repository

class dlkit.services.repository.**Repository**
Bases: dlkit.osid.objects.OsidCatalog, dlkit.osid.sessions.OsidSession

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

Asset Lookup Methods

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

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.

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.

Repository.**use_federated_repository_view**()
Federates the view for methods in this session. A federated view will include assets in repositories which are children of this repository in the repository hierarchy.

Repository.**use_isolated_repository_view**()
Isolates the view for methods in this session. An isolated view restricts lookups to this repository only.

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

`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

`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

`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

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

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

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

Asset Query Methods

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

Repository.**use_federated_repository_view**()

Federates the view for methods in this session. A federated view will include assets in repositories which are children of this repository in the repository hierarchy.

Repository.**use_isolated_repository_view**()
Isolates the view for methods in this session. An isolated view restricts lookups to this repository only.

Repository.**asset_query**
Gets an asset query.

Returns the asset query

Return type osid.repository.AssetQuery

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

Asset Admin Methods

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

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

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

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()

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

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

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()`

`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`

`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

`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`

`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

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

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

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

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()`

`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`

`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

`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()`

`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`

`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

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`

`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`

`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`

`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`

`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`

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`

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`

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`

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`

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

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

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

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

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

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

asset_content_ids
Gets the content Ids of this asset.

Returns the asset content Ids

Return type osid.id.IdList

asset_contents
Gets the content of this asset.

Returns the asset contents

Return type `osid.repository.AssetContentList`

Raise `OperationFailed` – unable to complete request

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`

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`

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

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`

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`

`title`
Sets the title.

Parameters **`title`** (`string`) – the new title

Raise `InvalidArgument` – `title` is invalid

Raise `NoAccess` – `Metadata.isReadOnly()` is `true`

Raise `NullArgument` – `title` is `null`

`public_domain_metadata`
Gets the metadata for the public domain flag.

Returns metadata for the public domain

Return type `osid.Metadata`

`public_domain`
Sets the public domain flag.

Parameters **`public_domain`** (`boolean`) – the public domain status

Raise `NoAccess` – `Metadata.isReadOnly()` is `true`

`copyright_metadata`
Gets the metadata for the copyright.

Returns metadata for the copyright

Return type `osid.Metadata`

`copyright_registration_metadata`
Gets the metadata for the copyright registration.

Returns metadata for the copyright registration

Return type `osid.Metadata`

`copyright_registration`
Sets the copyright registration.

Parameters **`registration`** (`string`) – the new copyright registration

Raise `InvalidArgument` – `copyright` is invalid

Raise `NoAccess` – `Metadata.isReadOnly()` is `true`

Raise `NullArgument` – `copyright` is `null`

`distribute_verbatim_metadata`
Gets the metadata for the distribute verbatim rights flag.

Returns metadata for the distribution rights fields

Return type `osid.Metadata`

`distribute_verbatim`
Sets the distribution rights.

Parameters **`distribute_verbatim`** (`boolean`) – right to distribute verbatim copies

Raise `InvalidArgument` – `distribute_verbatim` is invalid

Raise `NoAccess` – authorization failure

distribute_alterations_metadata
Gets the metadata for the distribute alterations rights flag.

Returns metadata for the distribution rights fields

Return type osid.Metadata

distribute_alterations
Sets the distribute alterations flag.

This also sets distribute verbatim to true.

Parameters **distribute_mods** (boolean) – right to distribute modifications

Raise InvalidArgument – distribute_mods is invalid

Raise NoAccess – authorization failure

distribute_compositions_metadata
Gets the metadata for the distribute compositions rights flag.

Returns metadata for the distribution rights fields

Return type osid.Metadata

distribute_compositions
Sets the distribution rights.

This sets distribute verbatim to true.

Parameters **distribute_comps** (boolean) – right to distribute modifications

Raise InvalidArgument – distribute_comps is invalid

Raise NoAccess – authorization failure

source_metadata
Gets the metadata for the source.

Returns metadata for the source

Return type osid.Metadata

source
Sets the source.

Parameters **source_id** (osid.id.Id) – the new publisher

Raise InvalidArgument – source_id is invalid

Raise NoAccess – Metadata.isReadOnly() is true

Raise NullArgument – source_id is null

provider_links_metadata
Gets the metadata for the provider chain.

Returns metadata for the provider chain

Return type osid.Metadata

provider_links
Sets a provider chain in order from the most recent source to the originating source.

Parameters **resource_ids** (osid.id.Id[]) – the new source

Raise InvalidArgument – resource_ids is invalid

Raise NoAccess – Metadata.isReadOnly() is true

Raise NullArgument – resource_ids is null

created_date_metadata
Gets the metadata for the asset creation date.

Returns metadata for the created date

Return type osid.Metadata

created_date
Sets the created date.

Parameters **created_date** (osid.calendaring.DateTime) – the new created date

Raise InvalidArgument – created_date is invalid

Raise NoAccess – Metadata.isReadOnly() is true

Raise NullArgument – created_date is null

published_metadata
Gets the metadata for the published status.

Returns metadata for the published field

Return type osid.Metadata

published
Sets the published status.

Parameters **published** (boolean) – the published status

Raise NoAccess – Metadata.isReadOnly() is true

published_date_metadata
Gets the metadata for the published date.

Returns metadata for the published date

Return type osid.Metadata

published_date
Sets the published date.

Parameters **published_date** (osid.calendaring.DateTime) – the new published date

Raise InvalidArgument – published_date is invalid

Raise NoAccess – Metadata.isReadOnly() is true

Raise NullArgument – published_date is null

principal_credit_string_metadata
Gets the metadata for the principal credit string.

Returns metadata for the credit string

Return type osid.Metadata

principal_credit_string
Sets the principal credit string.

Parameters **credit_string** (string) – the new credit string

Raise InvalidArgument – credit_string is invalid

Raise NoAccess – Metadata.isReadOnly() is true

Raise NullArgument – credit_string is null

composition_metadata
Gets the metadata for linking this asset to a composition.

Returns metadata for the composition

Return type osid.Metadata

composition
Sets the composition.

Parameters **composition_id** (osid.id.Id) – a composition

Raise InvalidArgument – composition_id is invalid

Raise NoAccess – Metadata.isReadOnly() is true

Raise NullArgument – composition_id is null

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

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

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

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

asset

Gets the Asset corresponding to this content.

Returns the asset

Return type osid.repository.Asset

accessibility_types

Gets the accessibility types associated with this content.

Returns list of content accessibility types

Return type osid.type.TypeList

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

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

data

Gets the asset content data.

Returns the length of the content data

Return type osid.transport.DataInputStream

Raise OperationFailed – unable to complete request

has_url()
Tests if a URL is associated with this content.

Returns true if a URL is available, false otherwise

Return type boolean

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

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

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

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

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

accessibility_types

data_metadata
Gets the metadata for the content data.

Returns metadata for the content data

Return type osid.Metadata

data
Sets the content data.

Parameters **data** (osid.transport.DataInputStream) – the content data

Raise InvalidArgument – data is invalid

Raise NoAccess – Metadata.isReadOnly() is true

Raise NullArgument – data is null

url_metadata
Gets the metadata for the url.

Returns metadata for the url

Return type osid.Metadata

url
Sets the url.

Parameters **url** (string) – the new copyright

Raise InvalidArgument – url is invalid

Raise NoAccess – Metadata.isReadOnly() is true

Raise NullArgument – url is null

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

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

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

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

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

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

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`

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

title_terms

match_public_domain(*public_domain*)
Matches assets marked as public domain.

Parameters **public_domain** (`boolean`) – public domain flag

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

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`

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

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`

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

copyright_registration_terms

match_distribute_verbatim(*distributable*)
Matches assets marked as distributable.

Parameters **distributable** (`boolean`) – distribute verbatim rights flag

distribute_verbatim_terms

match_distribute_alterations(*alterable*)
Matches assets that whose alterations can be distributed.

Parameters **alterable** (`boolean`) – distribute alterations rights flag

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

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`

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`

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`

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

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`

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

created_date_terms

match_published(*published*)
Marks assets that are marked as published.

Parameters **published** (`boolean`) – published flag

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`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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`

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

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`

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`

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`

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

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`

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

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

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

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

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

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

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

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

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 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.

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.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U