Open Service Interface Definition

Dictionary

Document Release: 2.0
OSID version 2.0

Summary

The Dictionary Open Service Interface Definition (OSID) supports creating and populating dictionaries of tag-value pairs[1]. Each Dictionary has a name and description as well as a domain for which the Dictionary is intended. Other Dictionaries may share the same name for a different domain. This Service can be used for localization as well as for mapping a set of values for one domain to another.

The Dictionary OSID potentially has uses in many other OSID implementations when values need to be returned that might vary with the implementation's context.

Service Definition

An examination of an Open Service Interface Definition (OSID) usually begins with the Manager. All Managers[2] provide the way to create the objects that implement the principal interfaces in the Service. Before discussing the Manager in detail, we will review the intended role of the Dictionary Service overall.

A Dictionary is a relatively simple mechanism. In place of defining a specific value to be used in all contexts, a value is associated with a tag[3]. References are always made to the tag, rather than the value associated with it, so that the value can change but the instructions referencing the tag do not[4]. A common form of dictionary is the properties or configuration file. Of course, the language dictionary is a well-known model for mapping the values in one context, a language, to another. Mappings do not always need to be one-to-one. For example, one might have a tag for the kinds of services one offers. This list would vary by context and provider. In a restaurant, what is served is different but the tag "menu" is always the same. In a learning system, the kind of courses offered might be such a list.

Each value is an object. This means it can be a String, a number, or a complex multi-element structure. Each value is identified unambiguously using a String tag and a domain. The domain is an object rather than a String so that it can contain as much information as is needed to identify the domain[5]. The Service

[1] In Java™ one could implement many of the Services by using an implementation of java.util.Dictionary such as java.util.HashTable or java.util.HashMap.

[2] org.osid.OsidManager defines the interface extended by the Managers in each OSID. Here we will be discussing org.osid.dictionary.DictionaryManager. Refer to OsidManager for more information.

[3] In this Service, for simplicity and convenience, the tag is a String not an object.

[4] This indirection has, at a minimum, the benefit of not requiring recompilation of source code when values change.

[5] In Java, one option for a domain is a java.util.Locale object.

creates separate dictionaries for each domain[6]. Entries, tag-value pairs, are added and removed without reference to the domain.[7]

The Service provides for getting values this way:

```
value1 = DomainSpecificDictionary1.getEntry(tag)
value2 = DomainSpecificDictionary2.getEntry(tag)
```

rather than this way:

```
value1 = MultiDomainDictionary.getEntry(tag,domain1)
value2 = MultiDomainDictionary.getEntry(tag,domain2)
```

osid.dictionary.DictionaryManager

The DictionaryManager's **createDictionary**() method creates a Dictionary with a specific name, description, and domain. An Id for the Dictionary is created by an implementation. The **getDictionary**() method gets a Dictionary by dictionaryId. It is reasonable to use the same name for Dictionaries that differ only by domain, but having unique Dictionary names is also acceptable. The **getDictionaries**() method returns all the Dictionaries. The domain associated with each Dictionary is returned by a property of the Dictionary. The **deleteDictionary**() method deletes the Dictionary which is identified unambiguously by Id. Note there are no Dictionary Types.

DictionaryManager Method Summary

Dictionary	createDictionary(String name, String description, Serializable domain)
void	deleteDictionary(Id dictionaryId)
DictionaryIterator	getDictionaries()
Dictionary	getDictionary(Id dictionaryId)

org.osid.dictionary.Dictionary

The Dictionary has a few property methods, **getDisplayName**(), **updateDisplayName**(), **getDescription**(), **updateDescription**(), **getDomain**(), and **getId**(). Note that in this Service, display names are mutable. As usual, Ids are immutable and are created by an implementation. Domains are also immutable. Each entry is added[8], using **addEntry**() and a tag-value pair. The tag, which is a String, needs to be unique within the Dictionary because the **removeEntry**() and **getEntry**() take a String not an Id.[9] The **getTags**() method returns all the tags.[10]

[6] An implementation could choose to implement a single underlying store for all domains and use the Dictionary as an additional key. One advantage in separate stores by domain is it may be easier to have decentralized localization efforts.

[7] Other Java approaches, for example some of those using java.util.Locale, always specify the domain when managing the tag.

[8] Implementations have the choice of using throwing an exception with an "already added" message if the tag is already defined or not throwing and exception and updating the value.

[9] There is no separate Entry interface with its own Id.

[10] The number of tags returned could be large. Also there is no support for searching the tags to return, for example, those matching some pattern. If that kind of support is desired, refer to OSID Repository.

Dictionary Method Summary

void	addEntry(String tag, Serializable value)
String	getDescription()
String	getDisplayName()
Serializable	getDomain()
Serializable	getEntry(String tag)
Id	getId()
StringIterator	getTags()
void	removeEntry(String tag)
void	updateDescription(String description)
void	updateDisplayName(String displayName)

org.osid.dictionary.DictionaryException

The service definitions make use of Exceptions as a mechanism for responding to error or unusual conditions. All methods in the Dictionary OSID throw a **DictionaryException**. The Exception contains a message that is a String.

The following message String is defined in DictionaryException:

- UNKNOWN_TAG

Note that other kinds of Exception constructors are not used as all do or can devolve to a String. All methods of all interfaces of all OSIDs throw a subclass of org.osid.OsidException. This requires the caller of any implementation method handle the Exception.

If a method performs an operation without incident, an object or primitive may be returned, but in most cases, methods do not return error codes or a success or failure boolean. For example, a method that deletes an object with a particular identifier, would throw an Exception if the identifier were unknown; the method would not return, for example, false.

org.osid.dictionary.DictionaryIterator

Iterators[11] return a set of elements of a specific type, one at a time. The purpose of Iterators is to offer a way for methods to return multiple values of a common type and not use an array. Returning an array may not be appropriate if the number of values returned is large or is fetched remotely. Iterators do not allow access to values by index, rather you must access values in sequence. Similarly, there is no way to go backwards through the sequence unless you place the values in a data structure, such as an array, that allows for access by index.

All iterators contain two methods. The **hasNext<*Object type*>**() method returns true if there are more values of the iterator type available; false otherwise. The **next<*Object type*>** () method returns the next element in the sequence. Note that in many cases, the order of elements is not guaranteed.

The following iterator is defined in this OSID:

- DictionaryIterator

[11] Refer to “Discussion of Iterators” for more information.