Open Service Interface Definition

Logging

Document Release: 2.0
OSID Release: 2.0

Summary

The Logging Open Service Interface Definition (OSID) provides a means of reading and writing entries to logs. The Service supports specified formats and priorities for each entry. Applications can use the Service to record activity for a production system while implementations of other OSIDs can use the Service to record detailed data during development, problem identification, performance, etc.

A given application can read or write to more than one log. Different applications can read or write to the same log using the same log name.

Service Definition

An examination of an Open Service Interface Definition (OSID) usually begins with the Manager. All Managers[1] 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 Logging Service overall.

Logging serves to record information. An application might call an implementation of the Service to record user activity of any kind. For example, application use statistics such as who was using the application and what they did, which application features were invoked, exceptions raised by implementations and handled by the application, and so on. This might all be information about a system in production. In this mode, logs might be written as well as read. Additionally, an implementation of an OSID or anything else might want to log information for debugging, problem identification, or performance data. In this mode, logs might only be written.

There are a variety of commercial logging applications and there is logging support in certain operating systems and development kits. OSID Logging is an abstraction that seeks to address a wide range of needs. The Service is general and focuses on reading from and writing to logs only.

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

org.osid.logging.LoggingManager

An installation may have several implementations of LoggingManager available. Like managers in others O.K.I. services, LoggingManagers with desired features can be obtained through OsidLoader. Once a LoggingManager is available, it can get the logs available for reading with **getLogNamesForReading**(). A pre-existing log can be accessed with **getLogNamesForWriting**(). If the log does not exist, use **createLog**() to create one. Note that logs do not have Ids. The name of the log should be unique for the implementation and if the name is already known, there is an exception message, DUPLICATE_NAME, which should be employed by the implementation. The **deleteLog**() method deletes a log by name. To get a specific pre-existing log for reading and writing use **getLogForReading**() and **getLogForWriting**(), respectively.

The Manager can also return the Format and Priority Types supported by the implementation. These are defined by the implementation and, as with all Types[2], their meaning needs to be agreed to outside of the OSID. Formats are intended to reflect the content and layout of entries, while Priorities are intended to stratify entries according to severity.

Method Summary

WritableLog	createLog (String logName)
void	deleteLog(String logName)
TypeIterator	getFormatTypes()
ReadableLog	getLogForReading(String logName)
WritableLog	getLogForWriting(String logName)
StringIterator	getLogNamesForReading()
StringIterator	getLogNameForWriting()
TypeIterator	getPriorityTypes()
boolean	supportsReading()

org.osid.Logging.ReadableLog

The ReadableLog is relatively basic. Logs can return their name for display. The *displayName* may or may not be identical to the *logName* used when it was originally created with **createWritableLog**().

Method Summary

String	getDisplayName()
EntryIterator	getEntries(Type formatType, Type priorityType)

org.osid.Logging.WritableLog

WritableLogs can be appended to. There are format and priority types that can be set and subsequently used by calls to **appendLog**() or these types can be specified with each call. There is no requirement that these settings persist beyond the lifetime of the current WritableLog instance..

Method Summary

void	appendLog(Serializable entryItem)
void	appendLogWithTypes(Serializable entryItem, Type formatType, Type priorityType)
String	getDisplayName()
void	assignFormatType(Type formatType)
void	assignPriorityType(Type priorityType)

[2] Refer to a discussion of Types in Shared OSID.

org.osid.Logging.Entry

The Entry simply holds a serializable object and the Types and timestamp.

Method Summary

Type	getFormatType()
Serializable	getItem()
Type	getPriorityType()
long	getTimeStamp()

org.osid.logging.EntryIterator

Iterators return a set of elements of a specific type, one at a time. The purpose of all Iterators is to offer a way for OSID 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.

Method Summary

boolean	hasNextEntry()
Entry	nextEntry()

org.osid.logging.LoggingException

The OSIDs make use of Exceptions as a mechanism for responding to error or unusual conditions. All methods in the Logging OSID throw a **LoggingException**. The Exception contains a message that is a String. The following message Strings are defined in LoggingException:

Exception Message Summary

Constant	**Message String**
UNKNOWN_NAME	Log with this name cannot be found
DUPLICATE_NAME	There is already a Log with this name
FORMAT_TYPE_NOT_SET	Default format Type has not been set
PRIORITY_TYPE_NOT_SET	Default priority Type has not been set

If an implementation uses these messages, consumers of the implementation can easily test and conditionally respond to the Exception. 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.