Open Service Interface Definition

SQL

Document Release: 2.0
OSID version: 2.0

Summary

The SQL Open Service Interface Definition (OSID) provides a means of storing, editing, and retrieving tabular data through the execution of SQL statements. The data store is presumed to be a relational database.

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 SQL Service overall.

Because of the variety of backend systems that an implementation may use to support the SQL OSID not all the methods may be implemented. Methods that are unimplemented due to the limitations of the backend system should throw a SqlException with the defined message, UNIMPLEMENTED.

org.osid.sql.SQLManager

An installation may have several implementations of SQLManager available. Like managers in other OKI services, SQLManagers with desired features can be obtained through OsidLoader. Once a SQLManager is available, it can create a connection to a database using the **getConnection()** method. SqlException may be thrown when arguments are null, methods have not been implemented, default SQLManager does not load, and the SQLManager property file does not load.

All implementors of an OSID provide create, delete, and get methods for the various objects defined in the package. Most also include methods for returning Types. We use create methods in place of the new operator. Create method implementations should both instantiate and persist objects. The reason we avoid the `new` operator is that it makes the name of the implementing package explicit and requires a source code change in order to use a different package name. In combination with OsidLoader, applications developed using OSIDs permit implementation substitution without source code changes.

SQLManager Method Summary

Connection	getConnection(String connectionString)

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

org.osid.sql.Blob

A Blob is a variable length representation of binary (byte) values. (Blob stands for Binary Large Object.) The contents of a Blob are returned in a ByteValueIterator by the **getBytes()** method. Its length is returned by the method length().

Blob Method Summary

ByteValueIterator	getBytes()
long	length()

org.osid.sql.Clob

A Clob is a variable length representation of character values. (Clob stands for Character Large Object.) The contents of a Clob are returned in a CharValueIterator by the **getChars()** method. Its length is returned by the method length().

Clob Method Summary

CharValueIterator	getChars()
long	length()

org.osid.sql.Connection

The Connection represents the channel of communication with the database. It is the means of submitting a query or update for execution. A ResultTable is returned by the **executeQuery()** methods. The **executeUpdate()** methods are used to execute commands that don't return resultsets such as inserts, deletes, and other DDL commands. The methods **executeQueryWithParams()** and **executeUpdateWithParams()** support passing parameters and parameter metadata with the SQL statement. You may retrieve the valid SQL data types defined by the implementation using the method **getSqlTypes()**.

Connection Method Summary

ResultTable	executeQuery(string sql)
ResultTable	executeQueryWithParams(string sql, Object[] objs)
int	executeUpdate(string sql)
int	executeUpdateWithParams(string sql, Object[] objs)
TypeIterator	getSqlTypes()

org.osid.sql.ResultMetaData

ResultMetaData contains information about a column in the ResultTable. You may retrieve the ResultMetaData for each column by calling the ResultTable method **getResultMetaData()**.

ResultMetaData contains a variety of methods to learn more about the column's data. The method **areNullsAllowed()** indicates whether the column may contain a null value. The method **getColumnIndex()** returns the columns ordinal value. You can determine the column's data type using the method **getColumnType()**. This method returns a Type that represents a predefined SQL data type. These Types may be the SQL data types specified by the database vendor, or those specified as part of the implementation. The method getTableName() returns the table, catalog, or schema name associated with the column's data.

ResultMetaData Method Summary

boolean	areNullsAllowed()
int	getColumnIndex()
String	getColumnName()
Type	getColumnType()

org.osid.sql.Row

A Row contains the row contents from a ResultTable. Its methods provide access to the data returned in a ResultTable. Individual columnar data may be retrieved by specifying either the column index or the column name using the methods **getColumnByIndex()** and **getColumnByName()**. You may retrieve all columns in a Row using the method **getColumns()** which returns an ObjectIterator.

Row Method Summary

Serializable	getColumnByIndex(int columnIndex)
Serializable	getColumnByName(String columnName)
ObjectIterator	getColumns()

org.osid.sql.ResultTable

A ResultTable is tabular data returned by a query. You access the returned data by calling the method **getRows()** which returns a RowIterator. Information about each ResultTable column is obtained through the method **getResultMetaData()** which returns a ResultMetaDataIterator. You may get the number of columns in the ResultTable by calling the method **getColumnCount().**

ResultTable Method Summary

int	getColumnCount()
ResultMetaDataIterator	getResultMetaData()
RowIterator	getRows()

org.osid.sql.ResultMetaDataIterator
org.osid.sql.RowIterator

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.

ResultMetaDataIterator Method Summary

boolean	hasNextResulMetaData()

ResultMetaData	nextResultMetaData()

RowIterator Method Summary

boolean	hasNexRowt()
Row	nextRow()

org.osid.sql.SqlException

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

Constant	Message String
BLOB_GETBYTES_FAILED	Blob get bytes failed
CLOB_GETCHARS_FAILED	Clob get chars failed
CONNECTION_FAILED	Connection failed
DATA_RETRIEVAL_ERRPR	Get row metadata failed
INVALID_ARGUMENTS	Invalid arguments
INVALID_COLUMN_INDEX	Invalid column index
OPERATION_FAILED	Operation failed
NO_MORE_ITERATOR_ELEMENTS	No more iterator elements
UNKNOWN_TYPE	Unknown Type

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 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.

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 osid.OsidException. This requires the caller of any implementation method handle the Exception.

SqlException Method Summary

SqlException has no defined methods.