OracleEOAdaptor Framework

Objective–C API Reference

OracleEOAdaptor Framework

Framework:	System/Library/Frameworks/OracleEOAdaptor.framework
Header File Directories:	System/Library/Frameworks/OracleEOAdaptor.framework/ Headers

Introduction

The OracleEOAdaptor framework is a set of classes that allow your programs to connect to an Oracle server. These classes provide Oracle-specific method implementations for the EOAccess framework's EOAdaptor, EOAdaptorChannel, EOAdaptorContext, and EOSQLExpression abstract classes.

The following table lists the classes in the OracleEOAdaptor Framework and provides a brief description of each class.

Class	Description
OracleAdaptor	Represents a single connection to a Oracle database server, and is responsible for keeping login and model information, performing Oracle-specific formatting of SQL expressions, and reporting errors.
OracleChannel	Represents an independent communication channel to the database server its OracleAdaptor is connected to.
OracleContext	Represents a single transaction scope on the database server to which its adaptor object is connected.
OracleSQLExpression	Defines how to build SQL statements for OracleChannels.

The Connection Dictionary

The connection dictionary contains items needed to connect to an Oracle server, such as the server name and database (it's common to omit the user name and password from the connection dictionary, and prompt users to enter those values in a login panel). The keys of this dictionary identify the information the server expects, and the values of those keys are the values that the adaptor uses when trying to connect to the server.

The Oracle adaptor defines string constants for use as connection dictionary keys:

- `ServerIdKey`

- `HostMachineKey`

- `UserNameKey`

- `PasswordKey`

- `ConnectionStringKey`

- `NlsLangKey`

The OracleAdaptor attempts to connect with a connection string of the form "userName/ password@serverId". If all the values except the one for `ServerIdKey` are absent, then OracleAdaptor attempts to connect with just the value for `ServerIdKey`.

The `ConnectionStringKey` and `NlsLangKey` are optional. The value for `ConnectionStringKey` is a string to be used to login to the database server. If the `ConnectionStringKey` is present in the connection dictionary, the other logon keys (`ServerIdKey`, `HostMachineKey`, `UserNameKey`, and `PasswordKey`) are ignored and the value for the `ConnectionStringKey` is used to connect to the database.

The value for `NlsLangKey` is used to set the Oracle NLS_LANG environment variable. NLS_LANG declares to the Oracle server the character set being used by the client, as well as the language in which you want server error messages to appear. The format is as follows:

```
language_territory. characterSet
```

For example, supplying the value `japanese_japan.jeuc` for the `NlsLangKey` tells the server that the language is Japanese, the territory is Japan, and the character set is jeuc. See your Oracle documentation for a complete list of types available for this field.

To add the `NlsLangKey` and a value to your connection dictionary, you can manually edit your model file. For example:

```
connectionDictionary = {
```

```
    password = tiger;
    serverId = sjOracle;
    userName = scott;
    NLS_LANG = american_america.us7ascii;
};
```

Subsequently changing the connection dictionary in your model file using the Set Adaptor Info command in EOModeler has no effect on these keys and their values-they are preserved unless you edit the file to remove them.

The default character set for Japanese systems is jeuc. If you are using a non-Japanese system, the default is whatever Oracle provides. You only need to add the `NlsLangKey` to your connection dictionary if you are using a character set other than your system's default.

> **Note:** Enterprise Objects Framework uses Mac OS X Server encoding to represent string data, and it passes strings to the database without converting them to the database character set. If you require that the data passed to your server is in an encoding other than Mac OS X Server encoding, you need to subclass NSString.

Locking

All adaptors use the database server's native locking facilities to lock rows on the server. The Oracle adaptor locks a row by using the SELECT... FOR UPDATE... statement. This occurs when:

- You send the adaptor channel a `selectAttributes:fetchSpecification:lock:entity:` message with `YES` specified as the value for the `lock:` keyword.

- You explicitly lock an object's row with the EODatabaseContext's `lockObjectWithGlobalID: editingContext:` message.

- You set pessimistic locking at the database level and fetch objects.

Data Type Mapping

Every adaptor provides a mapping between each server data type and the Objective-C type to which a database value will be coerced when it's fetched from the database. The following table lists the mapping used by OracleAdaptor.

Oracle Data Type	Objective-C Data Type
VARCHAR2	NSString
NUMBER	NSDecimalNumber
LONG	NSString
DATE	NSCalendarDate
RAW	NSData
LONG RAW	NSData
CHAR	NSString
MLSLABEL	NSString
REFCURSOR	OracleChannel

The type mapping methods—`externalTypesWithModel:`, `internalTypeForExternalType:model:`, and `isValidQualifierType:model:`—allow for an adaptor to supplement its set of type mappings with additional mappings for user-defined database types. OracleAdaptor does not make use of the model argument if one is provided.

Prototype Attributes

The OracleAdaptor Framework provides the following set of prototype attributes:

Name	External Type	Value Class Name	Other Attributes
binaryID	RAW	NSData	width = 12
city	VARCHAR2	NSString	columnName = CITY width = 50
date	DATE	NSCalendarDate	columnName = ""
longText	LONG	NSString	
money	NUMBER	NSDecimalNumber	columnName = ""
phoneNumber	VARCHAR2	NSString	columnName = PHONE width = 20
rawImage	LONG RAW	NSData	columnName = RAW_IMAGE
state	VARCHAR2	NSString	columnName = STATE width = 2
streetAddress	VARCHAR2	NSString	columnName = STREET_ADDRESS width = 100
tiffImage	LONG RAW	NSImage	adaptorValueConversionMethodName = TIFFRepresentation columnName = PHOTO valueFactoryMethodName = "imageWithData:"
uniqueID	NUMBER	NSNumber	columnName = "" valueType = i
zipCode	VARCHAR2	NSString	columnName = ZIP width = 10

Handling Errors

OracleChannel provides a method for handling errors: `raiseOracleError`. This method is invoked whenever the channel encounters an error reported by the Oracle server. The methods `cursorDataArea`, `hostDataArea`, and `logonDataArea` are used to retrieve Oracle-specific data structures from the channel and context to determine what error has occurred. The `hostDataArea` and `logonDataArea` methods are declared in the OracleContext class.

Generating Primary Keys

Each adaptor provides a database-specific implementation of the method `primaryKeyForNewRowWithEntity:` for generating primary keys. The OracleChannel's implementation uses sequence objects to provide primary key values. The statement used to create the sequence is:

```
create sequence table_SEQ
```

where table is the name of the table for which the adaptor provides primary key values. The adaptor sets the sequence start value to the corresponding table's maximum primary key value plus one.

To use OracleChannel's database-specific primary key generation mechanism, be sure that your database accommodates the adaptor's scheme. To modify your database so that it supports the adaptor's mechanism for generating primary keys, use EOModeler. For more information on this topic, see *Enterprise Objects Framework Developer's Guide*.

Bind Variables

The OracleAdaptor uses bind variables. A bind variable is a placeholder used in an SQL statement that is replaced with an actual value after the database server determines an execution plan. You use the following OracleSQLExpression methods to operate on bind variables:

■ bindVariableDictionaryForAttribute:value:

■ mustUseBindVariableForAttribute:

■ shouldUseBindVariableForAttribute:

OracleAdaptor

Inherits from:	EOAdaptor : NSObject
Declared in:	OracleEOAdaptor/OracleAdaptor.h

Class Description

An OracleAdaptor represents a single connection to an Oracle database server, and is responsible for keeping login and model information, performing Oracle-specific formatting of SQL expressions, and reporting errors.

The OracleAdaptor class has these restrictions: You can't have nested transactions, and the adaptor doesn't support full outer joins.

Constants

OracleAdaptor defines the following string constants for use as connection dictionary keys.

Constant	Corresponding value in the connection dictionary
ServerIdKey	The server ID. Used as the string to connect to the database if values for HostMachineKey, UserNameKey, and PasswordKey are not present in the database.
HostMachineKey	The name of the host machine. If this key is not present, the string used to log in to the database is of the form "userName/password@serverId".
UserNameKey	The name of the user to log in as.
PasswordKey	The user's password.
ConnectionStringKey	The connection string used to log in to a database server. If this key is present, ServerIdKey, HostMachineKey, UserNameKey, and PasswordKey are ignored.
NlsLangKey	The setting to NLS_LANG, which is used to specify the language and character set for server connections. On J systems this option defaults to japanese_japan.jeuc.

See <u>"The Connection Dictionary"</u> (page 4) in the OracleEOAdaptor framework introduction for more information on the connection dictionary and its entries.

OracleAdaptor also defines a string constant for use as a key in an exception's userInfo dictionary (see raiseOracleError in the OracleChannel class specification).

Constant	Corresponding value in an exception's userInfo dictionary
OracleErrorKey	The Oracle OCI client library error code. The value is an NSNumber with the error code as its intValue, a positive number.

Instance Methods

connectionKeys

```
– (NSArray *)connectionKeys
```

Returns an NSArray containing the keys in the receiver's connection dictionary. You can use this method to prompt the user to supply values for the connection dictionary.

fetchedValueForDateValue:attribute:

```
– (NSCalendarDate *)fetchedValueForDateValue:(NSCalendarDate *)date
   attribute:(EOAttribute *)attribute
```

Returns an NSCalendarDate based on date whose millisecond value is set to 0.

fetchedValueForNumberValue:attribute:

```
– (NSNumber *)fetchedValueForNumberValue:(NSNumber *)numberValue
   attribute:(EOAttribute *)attribute
```

Returns an NSNumber based on *numberValue* that has been rounded according to the precision and scale specified for attribute.

oracleConnectionString

```
– (NSString *)oracleConnectionString
```

Returns the user name, password, host machine, and server id as a string suitable to be supplied as an argument to `orlon()`.

OracleChannel

Inherits from:	EOAdaptorChannel : NSObject
Declared in:	OracleEOAdaptor/OracleChannel.h
	OracleEOAdaptor/OracleDescription.h

Class Description

An OracleChannel represents an independent communication channel to the database server its OracleAdaptor is connected to. All of an OracleChannel's operations take place within the context of transactions controlled or tracked by its OracleContext. An OracleContext can manage multiple OracleChannels, and a channel is associated with only one context.

The features OracleChannel adds to EOAdaptorChannel are as follows:

■ Oracle-specific error handling

■ The ability to configure the fetch buffer

■ The ability to read a default list of table names from the database

Method Types

Finding table names

+ oracleTableNamesSQL

+ setOracleTableNamesSQL:

– describeTableNames

Accessing the fetch buffer length

– fetchBufferLength

– setFetchBufferLength:

Error handling

– raiseOracleError

– cursorDataArea

Class Methods

oracleTableNamesSQL

+ (NSString *)oracleTableNamesSQL

Returns the SQL statement that will be executed when building a default model.

setOracleTableNamesSQL:

+ (void)setOracleTableNamesSQL:(NSString *)*sql*

Sets to *sql* the SQL statement that will be used to return a list of table names from the database. By default, this list is the result of the SQL statement:

```
SELECT TABLE_NAME FROM USER_TABLES ORDER BY TABLE_NAME
```

This setting is used by all OracleChannels in an application. You can specify a different SQL statement using the defaults write command, for example:

```
% defaults write NSGlobalDomain OracleTableNamesSQL "SELECT TABLE_NAME FROM..."
```

Once you use `setOracleTableNamesSQL:` to specify a setting, it supersedes values set with the `defaults write` command.

Instance Methods

cursorDataArea

```
- (struct cda_def *)cursorDataArea
```

If the channel is connected, returns an Oracle-specific data structure (`cda_def`) describing characteristics of the channel. Otherwise, returns `NULL`. This method is commonly used with the method `raiseOracleError` to determine why an error occurred.

describeTableNames

```
- (NSArray *)describeTableNames
```

Overrides the EOAdaptorChannel implementation to return an array of the names of all the tables owned by the current user. Uses the SQL returned by `oracleTableNamesSQL`.

fetchBufferLength

```
- (unsigned)fetchBufferLength
```

Returns the size, in bytes, of the fetch buffer. The larger the buffer, the more rows can be returned for each round trip to the server.

raiseOracleError

```
– (void)raiseOracleError
```

Examines Oracle structures for error flags and raises an exception if one is found. The userInfo of the exception contains the Oracle error code in an entry with the key `OracleErrorKey`.

setFetchBufferLength:

```
– (void)setFetchBufferLength:(unsigned)length
```

Sets to *length* the size, in bytes, of the fetch buffer. The larger the buffer, the more rows can be returned for each round trip to the server.

OracleContext

Inherits from:	EOAdaptorContext : NSObject
Declared in:	OracleEOAdaptor/OracleContext.h

Class Description

An OracleContext represents a single transaction scope on the database server to which its adaptor object is connected. If the server supports multiple concurrent transaction sessions, the adaptor may have several adaptor contexts. An OracleContext may in turn have several OracleChannels, which handle actual access to the data on the server.

The features the OracleContext class adds to EOAdaptorContext are methods for returning Oracle-specific data structures that describe characteristics of the context: hostDataArea and logonDataArea. hostDataArea returns the `hda_def` data structure, and logonDataArea returns the `lda_def` data structure. If you intend to extend the OracleContext by making calls to the Oracle API, you'll need these data structures.

Method Types

Managing a connection to the server

```
– connect
```

– disconnect

– isConnected

Tracking fetches

– fetchesInProgress

Returning Oracle data structures

– hostDataArea

– logonDataArea

Instance Methods

connect

– (void)connect

Opens a connection to the database server. OracleChannel sends this message to OracleContext when it (OracleChannel) is about to open a channel to the server.

disconnect

– (void)disconnect

Closes a connection to the database server. OracleChannel sends this message to OracleContext when it (OracleChannel) has just closed a channel to the server.

fetchesInProgress

– (unsigned)fetchesInProgress

Returns the number of fetches the receiver has in progress.

hostDataArea

```
– (unsigned char *)hostDataArea
```

If the channel is connected, returns an Oracle-specific data structure describing characteristics of the context. Otherwise, returns NULL. This method is commonly used with the OracleChannel method raiseOracleError to determine why an error occurred.

isConnected

```
– (BOOL)isConnected
```

Returns YES if the receiver has an open connection to the database, NO otherwise.

logonDataArea

```
– (void *)logonDataArea
```

If the channel is connected, returns an Oracle-specific data structure describing characteristics of the context. Otherwise, returns NULL. This method is commonly used with the OracleChannel method raiseOracleError to determine why an error occurred.

OracleSQLExpression

Inherits from:	EOSQLExpression : NSObject
Declared in:	OracleEOAdaptor/OracleSQLExpression.h

Class Description

OracleSQLExpression defines how to build SQL statements for OracleChannels.

Class Methods

serverTypeIdForName:

```
+ (int)serverTypeIdForName:(NSString *)typeName
```

Returns the Oracle type code (such as OraVARCHAR2 or OraNumber) for *typeName* (such as "VARCHAR2" or "NUMBER").

setUseNoWaitLocks:

`+ (void)setUseNoWaitLocks:(BOOL)`*`flag`*

Sets according to *flag* whether the lock clause of the OracleSQLExpression is @"FOR UPDATE" (block until the row is available) or @"FOR UPDATE NOWAIT" (return an error immediately if an attempt to lock a row would block). By default OracleSQLExpression uses the clause @"FOR UPDATE"—that is, by default it does not use NOWAIT locks. This behavior is also controllable through the `EOOracleUseNoWaitLocks` user default.

setUseQuotedExternalNames:

`+ (void)setUseQuotedExternalNames:(BOOL)`*`flag`*

Sets according to *flag* whether the OracleSQLExpression expects external (database) names to be enclosed in quotation marks. This is useful if the database has table or column names that are either reserved words or that are not all uppercase. The default is NO.

This behavior can also be controlled through the `EOOracleUseQuotedExternalNames` user default.

useNoWaitLocks

`+ (BOOL)useNoWaitLocks`

Returns `YES` to indicate that the OracleSQLExpression uses NOWAIT locks, `NO` otherwise. The default is `NO`.

useQuotedExternalNames

`+ (BOOL)useQuotedExternalNames`

Returns YES to indicate that the OracleSQLExpression uses quoted external (database) names, NO otherwise.

Instance Methods

lockClause

```
– (NSString *)lockClause
```

Overrides the EOSQLExpression method `lockClause` to return the SQL string used in a SELECT statement to lock selected rows. Queries the user default EOOracleUseNoWaitLocks. If this default is not set or if it is set to `NO`, this method returns the string @"FOR UPDATE". If the default is set to `YES`, this method returns @"FOR UPDATE NOWAIT".