EOAdaptorChannel

Inherits From: NSObject

Conforms To: NSObject (NSObject)

Declared In: EOAccess/EOAdaptorChannel.h

Class at a Glanceÿ

Purpose

EOAdaptorChannel is an abstract class that defines how an Enterprise Objects Framework application performs operations on a database server. Concrete subclasses of EOAdaptorChannel override many of its methods in terms of the client libraries for a specific RDBMS. You typically don't interact with adaptor channels directly; rather, the Enterprise Objects Framework creates instances of a concrete adaptor channel subclass automatically and handles all the necessary interactions with those instances. If you're not creating a concrete adaptor channel subclass, there aren't very many methods you need to use, and you'll rarely invoke them directly.

Principle Attributes

- Adaptor context
- Delegate

Creation

Other framework classes create them.	Adaptor channels are generally created automatically.
– createAdaptorChannel (EOAdaptorContext)	Creates an adaptor channel and assigns its context.

Commonly Used Methods

– openChannel	Opens the channel so it can perform database operations.
– closeChannel	Close the channel.
– selectAttributes:fetchSpecification:lock:entity:	Selects rows matching the specified qualifier.
– fetchRowWithZone:	Fetches a row resulting from the last **select...**, **executeStoredProcedure...**, or **evaluateExpression:**.
– insertRow:forEntity:	Inserts the specified row.
– updateValues:inRowDescribedByQualifier:entity:	Updates the row described by the specified qualifier.
– deleteRowDescribedByQualifier:entity:	Deletes the row described by the specified qualifier.
– executeStoredProcedure:withValues:	Performs the specified stored procedure.
– evaluateExpression:	Sends the specified expression to the database.

– performAdaptorOperation:	Performs an adaptor operation by invoking the EOAdaptorChannel method appropriate for performing the specified operation.

Class Description

EOAdaptorChannel is an abstract class that provides concrete subclasses with a structure for performing database operations. A concrete subclass of EOAdaptorContext provides database-specific method implementations and represents an independent communication channel to the database server to which its EOAdaptor object is connected. You never interact with instances of the EOAdaptorChannel class, rather your Enterprise Objects Framework applications use instances of concrete subclasses that are written to interface with a specific database or other persistent storage system. To create an instance of a concrete EOAdaptorChannel subclass, you send a **createAdaptorChannel** message to an instance of the corresponding EOAdaptorContext subclass. You rarely create adaptor channels yourself. They are generally created automatically by other framework objects.

You use an adaptor channel to manipulate rows (records) by selecting, fetching, inserting, deleting, and updating them. An adaptor channel also gives you access to some of the metadata on the server, such as what stored procedures exist, what tables exist, and what their basic attributes and relationships are.

All of an adaptor channel's operations take place within the context of transactions controlled or tracked by its EOAdaptorContext. An adaptor context may manage several channels (though not all can), but a channel is associated with only one context.

Connecting to the Database

Before you can begin a transaction or perform a database operation, you need to form a connection to the database server. An EOAdaptorContext holds the connection, but it isn't actually formed until one of the context's channels is *opened*. Using the EOAdaptorChannel method **openChannel**, you prepare the adaptor channel, its adaptor context, and its adaptor to communicate with the database server. If the adaptor channel is the first of a context's channels to be opened, preparing to communicate entails creating a database connection. Whether or not a channel's adaptor context is holding a database connection, you can't use an adaptor channel to interact with the server until you open the channel.

When you're done performing operations, you close an adaptor channel by sending a **closeChannel** message to the channel. An adaptor channel's context can have many channels. The connection to the database isn't actually closed until the last open channel is closed.

Performing Database Operations

An adaptor channel operates on database rows represented by NSDictionary objects. The keys of the dictionary represent the names of attributes (columns), the values for those keys are the values for the attributes. It's important to note that the table and column names in the database server are not the same as the entity and attribute names that the Enterprise Objects Framework uses (though an application can set them to be identical).

Once you've set up a suite of adaptor objects, opened the adaptor channel, and begun a transaction, you can use the adaptor channel to operate on data in the server. To insert a new row for a particular entity, for example, you create a dictionary object containing all the key-value pairs for the entity's properties, and send the channel an **insertRow:forEntity:** message.

The other database operations—delete and update—require an EOQualifier object to specify which rows to affect. An EOQualifier describes specific rows for a single entity based on the values of properties; for example, rows whose salary attribute is greater than 40,000. See the EOQualifier class specification for more information on creating qualifiers.

Fetching rows with an adaptor channel is a two-stage process. First, you select the rows with a **selectAttributes:fetchSpecification:lock:entity:** message. This establishes one or more result sets of records to be fetched. If the selection succeeds, you then send **fetchRowWithZone:** messages to retrieve individual rows until there are no more. This code template displays the highlights of the process:

```
EOAdaptorChannel *myChannel;            /* Assume this exists. */
EOEntity *employeeEntity;               /* Assume this exists. */
EOFetchSpecification *myFetchSpec;      /* Assume this exists. */
NSDictionary *theRow;

NS_DURING

if (![myChannel isOpen]) {
    [myChannel openChannel];
}


[myChannel selectAttributes:[employeeEntity attributes]
            fetchSpecification:myFetchSpec
            lock:NO
            entity:employeeEntity];

while (theRow = [myChannel fetchRowWithZone:NULL]) {
    /* Process theRow. */
}


NS_HANDLER
    /* Handle the exception. */
NS_ENDHANDLER
```

The adaptor layer classes—EOAdaptor, EOAdaptorContext, and EOAdaptorChannel—notify you when an error occurs by raising an exception, particularly when the error can occur in the database server. Therefore, the **openChannel**, **selectAttributes:fetchSpecification:lock:entity:**, and **fetchRowWithZone:** messages above can all raise exceptions if an error occurs. The code between the NS_HANDLER and NS_ENDHANDLER macros should find out what exception was raised and handle it appropriately. See the NSException class specification for more information on handling exceptions.

After you open the adaptor channel and begin a transaction, you use the method **selectAttributes:fetchSpecification:lock:entity:** to select records in preparation for fetching. This method requires you to specify the attributes you're interested in and a fetch specification. You can ask the EOEntity object that represents a table for any of its attributes by name, or you can ask it for all its attributes as the excerpt does above. The **fetchSpecification:** argument is an EOFetchSpecification object that provides a qualifier, a fetch order, and other specifications for the select. See the EOFetchSpecification class specification for more information on creating fetch specifications. Once you've selected a set of records, you use a loop to fetch each one individually with **fetchRowWithZone:**.

Getting Basic Schema Information

You can use the adaptor-level objects without loading an EOModel file by having the adaptor channel read a minimal model from the database server's system dictionary. To build such a model you use the "describe" methods:

– describeTableNames
– describeModelWithTableNames:
– describeStoredProcedureNames

describeTableNames reads and returns an array of table names from the database. **describeModelWithTableNames:** constructs a default model out of the database's metadata, and assigns the adaptor name and connection dictionary to the new model. You generally invoke **describeModelWithTableNames:** with the array of table names (or a subset) returned from **describeTableNames**. **addStoredProceduresNamed:toModel:** reads an array of stored procedure names from the database.

Sending SQL Statements Directly to the Server

An EOAdaptorChannel allows you to send SQL directly to the database server with its **evaluateExpression:** method. This gives you access to database-specific features not made available by the adaptor channel itself, but can cause state changes on the server that aren't noted by objects in your application. If you cause state changes in evaluating SQL, you're responsible for notifying the appropriate objects of the changes; the classes of the Enterprise Objects Framework that are affected by evaluation of SQL typically describe the kinds of notifications they require.

If the expression you evaluate returns results, you fetch them with the method **fetchRowWithZone:** as you would had you used **selectAttributes:fetchSpecification:lock:entity:**. If your adaptor doesn't have a model set, you can use **describeResults** to determine the attributes selected by the expression. Even if you

perform the selection by sending raw SQL to the server, this method returns usable attributes that allow you to fetch rows. The code snippet below outlines the steps:

```
EOAdaptorChannel *channel;              /* Assume this exists and it's open */
NSString *sqlString;                    /* Assume this exists */
EOAdaptor *adaptor;
EOSQLExpression *expression;

adaptor = [[channel adaptorContext] adaptor];
expression = [[adaptor expressionClass] expressionForString:sqlString];

NS_DURING

NSMutableArray *array = [NSMutableArray array];
NSDictionary *row;

[channel evaluateExpression:expression];

do {
   [channel setAttributesToFetch:[channel describeResults]];
   while (row = [channel fetchRowWithZone:nil]) {
        /* Process the row. */
   }
} while ([channel isFetchInProgress]); /* Loop to process the next result set. */

[[myChannel adaptorContext] commitTransaction];

NS_HANDLER
   /* Handle the exception. */
NS_ENDHANDLER
```

Some expressions may return multiple result sets in which rows have different types and numbers of attributes. **fetchRowWithZone:** returns **nil** if there are no more rows in the current result set, but there may be more result sets to fetch from. The **while** statement containing the **isFetchInProgress** message ensures that the code will continue fetching through all result sets. When a select statement results in multiple result sets, you must set the channel's **attributesToFetch** so it knows how to construct the fetched row. The attributes returned from **describeResults** are appropriate for use with **setAttributesToFetch:**.

EOAdaptorChannel also provides a method for executing stored procedures: **executeStoredProcedure:withValues:**. Similar to **evaluateExpression:**, you can fetch rows resulting from a stored procedure using **fetchRowWithZone:**. In addition, you can get stored procedure return values and in/out parameters with the method **returnValuesForLastStoredProcedureInvocation**.

Notifying the Adaptor Channel's Delegate

You can assign a delegate to an adaptor channel. The EOAdaptorChannel sends certain messages directly to the delegate, and the delegate responds to these messages on the channel's behalf. Many of the adaptor

channel methods notify the channel's delegate before and after an operation is performed. Some delegate methods, such as **adaptorChannel:shouldEvaluateExpression:**, let the delegate determine whether the channel should perform an operation. Others, such as **adaptorChannel:didEvaluateExpression:**, are simply notifications that an operation has occurred. The delegate has an opportunity to respond by implementing the delegate methods. If the delegate wants to intervene, it implements **adaptorChannel:shouldEvaluateExpression:**. If it simply wants notification when a transaction has begun, it implements **adaptorChannel:didEvaluateExpression:**.

Creating an EOAdaptorContext Subclass

EOAdaptorContext provides many default method implementations that are sufficient for concrete subclasses:

– adaptorContext
– delegate
– deleteRowDescribedByQualifier:entity:
– isDebugEnabled
– lockRowComparingAttributes:entity:qualifier:snapshot:
– performAdaptorOperation:
– performAdaptorOperations:
– updateValues:inRowDescribedByQualifier:entity:

The following methods establish structure and conventions that other Enterprise Objects Framework classes depend on and should be overridden with caution:

– dictionaryWithObjects:forAttributes:zone:
– initWithAdaptorContext:
– setDebugEnabled:
– setDelegate:

If you override any of the above methods, your implementations should incorporate the superclass's implementation through a message to **super**.

The remaining EOAdaptorChannel methods must be overridden by concrete subclasses in terms of the persistent storage system to which it interfaces:

– attributesToFetch
– cancelFetch
– closeChannel
– deleteRowsDescribedByQualifier:entity:
– describeModelWithTableNames:
– describeResults
– describeStoredProcedureNames
– describeTableNames
– evaluateExpression:
– executeStoredProcedure:withValues:

– fetchRowWithZone:
– insertRow:forEntity:
– isFetchInProgress
– isOpen
– openChannel
– primaryKeyForNewRowWithEntity:
– returnValuesForLastStoredProcedureInvocation
– selectAttributes:fetchSpecification:lock:entity:
– setAttributesToFetch:
– updateValues:inRowsDescribedByQualifier:entity:

Method Types

Creating an EOAdaptorChannel	– initWithAdaptorContext:
Getting the adaptor context	– adaptorContext
Opening and closing a channel	– openChannel – closeChannel – isOpen
Modifying rows	– insertRow:forEntity: – updateValues:inRowDescribedByQualifier:entity: – updateValues:inRowsDescribedByQualifier:entity: – deleteRowDescribedByQualifier:entity: – deleteRowsDescribedByQualifier:entity:
Fetching rows	– selectAttributes:fetchSpecification:lock:entity: – describeResults – setAttributesToFetch: – attributesToFetch – fetchRowWithZone: – dictionaryWithObjects:forAttributes:zone: – cancelFetch – isFetchInProgress
Invoking stored procedures	– executeStoredProcedure:withValues: – returnValuesForLastStoredProcedureInvocation
Assigning primary keys	– primaryKeyForNewRowWithEntity:
Sending SQL to the server	– evaluateExpression:
Batch processing operations	– performAdaptorOperation: – performAdaptorOperations:

Getting schema information	– describeTableNames – describeStoredProcedureNames – addStoredProceduresNamed:toModel: – describeModelWithTableNames:
Debugging	– setDebugEnabled: – isDebugEnabled
Setting the delegate	– delegate – setDelegate:

Instance Methods

adaptorContext

– (EOAdaptorContext *)**adaptorContext**

Returns the receiver's EOAdaptorContext. A subclass of EOAdaptorChannel doesn't need to override this method.

See also: – **initWithAdaptorContext:**

addStoredProceduresNamed:toModel:

– (void)**addStoredProceduresNamed:**(NSArray *)*storedProcedureNames*
toModel:(EOModel *)*model*

Overridden by subclasses to add *storedProcedureNames* to *model*. This method is used in conjunction with **describeStoredProcedureNames** to build a default model in EOModeler. Raises an exception if an error occurs.

attributesToFetch

– (NSArray *)**attributesToFetch**

Implemented by subclasses to return the set of attributes to retrieve with **fetchRowWithZone:**. An adaptor channel subclass should override this method without invoking EOAdaptorChannel's implementation.

See also: – **setAttributesToFetch:**

cancelFetch

– (void)**cancelFetch**

Implemented by subclasses to clear all result sets established by the last **selectAttributes:fetchSpecification:lock:entity:**, **executeStoredProcedure:withValues:**, or **evaluateExpression:** message and terminate the current fetch, so that **isFetchInProgress** returns NO.

An adaptor channel subclass should override this method without invoking EOAdaptorChannel's implementation.

closeChannel

– (void)**closeChannel**

Implemented by subclasses to close the EOAdaptorChannel so that it can't perform operations with the server. Any fetch in progress is canceled. If the receiver is the last open channel in an adaptor context and if the channel's adaptor context has outstanding transactions, closing the channel has server-dependent results: some database servers roll back all outstanding transactions but others do nothing. Regardless of whether outstanding transactions are rolled back, this method has the side effect of closing the receiver's adaptor context's connection with the database if the receiver is its adaptor context's last open channel.

An adaptor channel subclass should override this method without invoking EOAdaptorChannel's implementation.

See also: – **cancelFetch**, – **transactionNestingLevel** (EOAdaptorContext)

delegate

– (id)**delegate**

Returns the receiver's delegate. A subclass of EOAdaptorChannel doesn't need to override this method.

See also: – **setDelegate:**

deleteRowDescribedByQualifier:entity:

– (void)**deleteRowDescribedByQualifier:**(EOQualifier *)*qualifier* **entity:**(EOEntity *)*entity*

Deletes the row described by *qualifier*. Invokes **deleteRowsDescribedByQualifier:entity:** and raises an exception unless exactly one row is deleted. A subclass of EOAdaptorChannel doesn't need to override this method.

deleteRowsDescribedByQualifier:entity:

– (unsigned)**deleteRowsDescribedByQualifier:**(EOQualifier *)*qualifier* **entity:**(EOEntity *)*entity*

Implemented by subclasses to delete the rows described by *qualifier* and return the number of rows deleted. Raises an exception on failure. Some possible reasons for failure are:

- The adaptor channel isn't open
- The adaptor channel is in an invalid state (for example, it's fetching).
- An error occurs in the database server

An adaptor channel subclass should override this method without invoking EOAdaptorChannel's implementation.

See also: – **deleteRowDescribedByQualifier:entity:**, – **isOpen**, – **isFetchInProgress**, – **transactionNestingLevel** (EOAdaptorContext)

describeModelWithTableNames:

– (EOModel *)**describeModelWithTableNames:**(NSArray *)*tableNames*

Overridden by subclasses to create and return a default model containing entities for the tables specified in *tableNames*. Assigns the adaptor name and connection dictionary to the new model. This method is typically used in conjunction with **describeTableNames** and **describeStoredProcedureNames**.

EOAdaptorChannel's implementation does nothing. An adaptor channel subclass should override this method to create a default model from the database's metadata.

describeResults

– (NSArray *)**describeResults**

Implemented by subclasses to return an array of EOAttributes describing the properties available in the current result set, as determined by **selectAttributes:describedByQualifier:fetchOrder:lock:**, **executeStoredProcedure:withValues:**, or a statement evaluated by **evaluateExpression:**.

An adaptor channel subclass should override this method without invoking EOAdaptorChannel's implementation.

describeStoredProcedureNames

– (NSArray *)**describeStoredProcedureNames**

Overridden by subclasses to read and return an array of stored procedure names from the database. This method is used in conjunction with **addStoredProceduresNamed:toModel:** to build a default model in EOModeler. Raises an exception if an error occurs.

describeTableNames

– (NSArray *)**describeTableNames**

Overridden by subclasses to read and return an array of table names from the database. This method in conjunction with **describeModelWithTableNames:** is used to build a default model.

EOAdaptorChannel's implementation simply returns **nil**. An adaptor channel subclass should override this method to construct an array of table names from database metadata.

dictionaryWithObjects:forAttributes:zone:

– (NSMutableDictionary *)**dictionaryWithObjects:**(id *)*objects*
forAttributes:(NSArray *)*attributes*
zone:(NSZone *)*zone*

Used by EOAdaptorChannel subclasses to create dictionaries that can be returned from **fetchRowWithZone:**. Use of this method is optional but strongly recommended; it enables performance optimizations. A subclass of EOAdaptorChannel shouldn't override it.

evaluateExpression:

– (void)**evaluateExpression:**(EOSQLExpression *)*expression*

Implemented by subclasses to send *expression* to the database server for evaluation, beginning a transaction first and committing it after evaluation if a transaction isn't already in progress. Raises an exception if an error occurs. An EOAdaptorChannel uses this method to send SQL expressions to the database.

If *expression* results in a select operation being performed, you can fetch the results as you would if you had sent a **selectAttributes:fetchSpecification:lock:entity:**. Use the method **setAttributesToFetch:** at the completion of one result set, before you begin fetching the next.

evaluateExpression: invokes the delegate methods **adaptorChannel:shouldEvaluateExpression:** and **adaptorChannel:didEvaluateExpression:**.

An adaptor channel subclass should override this method without invoking EOAdaptorChannel's implementation. Note, however, that the upper layers of the Framework never invoke **evaluateExpression:** directly. Thus, adaptors for data stores that don't naturally support an expression language (for example, flat file adaptors) don't need to implement this method to work with the Framework.

See also: – **fetchRowWithZone:**

executeStoredProcedure:withValues:

– (void)**executeStoredProcedure:**(EOStoredProcedure *)*storedProcedure*
withValues:(NSDictionary *)*values*

Implemented by subclasses to execute *storedProcedure*. Any arguments to the stored procedure are in *values*. Use **fetchRowWithZone:** to get result rows and **returnValuesForLastStoredProcedureInvocation** to get return arguments and result status, if any. Raises an exception if an error occurs.

An adaptor channel subclass should override this method without invoking EOAdaptorChannel's implementation. Note, however, that the upper layers of the Framework never invoke **executeStoredProcedure:withValues:** directly. Thus, adaptors for data stores that don't support stored procedures (for example, flat file adaptors) don't need to implement this method to work with the Framework

fetchRowWithZone:

– (NSMutableDictionary *)**fetchRowWithZone:**(NSZone *)*zone*

Implemented by subclasses to fetch the next row from the result set of the last **selectAttributes:fetchSpecification:lock:entity:**, **executeStoredProcedure:withValues:**, or **evaluateExpression:** message sent to the receiver. Returns values for the receiver's **attributesToFetch**. When there are no more rows in the current result set, this method returns **nil**, and invokes the delegate method **adaptorChannelDidChangeResultSet:** if there are more results sets. When there are no more rows or result sets, this method returns **nil**, ends the fetch, and invokes **adaptorChannelDidFinishFetching:**. **isFetchInProgress** returns YES until the fetch is canceled or until this method exhausts all result sets and returns **nil**. This method also invoke the delegate methods **adaptorChannelWillFetchRow:** and **adaptorChannel:didFetchRow:**. Raises an exception if an error occurs.

An adaptor channel subclass should override this method without invoking EOAdaptorChannel's implementation.

See also: – **setAttributesToFetch:**

initWithAdaptorContext:

– **initWithAdaptorContext:**(EOAdaptorContext *)*adaptorContext*

The designated initializer for the EOAdaptorChannel class, this method is overridden by subclasses to initialize a newly allocated EOAdaptorChannel subclass and retain *adaptorContext*. Returns **self**.

You never invoke this method directly unless you are implementing a concrete adaptor context. It is invoked automatically from **createAdaptorChannel—**the EOAdaptorContext method you use to create a new adaptor channel.

A subclass of EOAdaptorChannel doesn't need to override this method, but may override it to perform additional initialization. A subclass that does override this method must incorporate the superclass's version through a message to **super**.

See also: – **adaptorContext**

insertRow:forEntity:

– (void)**insertRow:**(NSDictionary *)*row* **forEntity:**(EOEntity *)*entity*

Implemented by subclasses to insert the values of *row* into the table in the database that corresponds to *entity. row* is an NSDictionary whose keys are attribute names and whose values are the values to insert. Raises an exception on failure. Some possible reasons for failure are:

- The user logged in to the database doesn't have permission to insert a new row.
- The adaptor channel is in an invalid state (for example, fetching).
- The row fails to satisfy a constraint defined in the database server.

An adaptor channel subclass should override this method without invoking EOAdaptorChannel's implementation.

isDebugEnabled

– (BOOL)**isDebugEnabled**

Returns YES if the adaptor channel logs evaluated SQL and other useful information to the console (or to the standard error stream), NO if not. A subclass of EOAdaptorChannel doesn't need to override this method.

See also: – **setDebugEnabled:**, – **setDebugEnabled:** (EOAdaptorContext)

isFetchInProgress

– (BOOL)**isFetchInProgress**

Implemented by subclasses to return YES if the receiver is fetching, NO otherwise. An adaptor channel is fetching if:

- It's been sent a successful **selectAttributes:describedByQualifier:fetchOrder:lock:** message.
- A stored procedure that returns rows has been successfully executed using **executeStoredProcedure:withValues:**.
- An expression sent through **evaluateExpression:** resulted in a select operation being performed.

An adaptor channel stops fetching when there are no more records to fetch or when it's sent a **cancelFetch** message.

An adaptor channel subclass should override this method without invoking EOAdaptorChannel's implementation.

See also: – **fetchRowWithZone:**

isOpen

– (BOOL)**isOpen**

Implemented by subclasses to return YES if the channel has been opened with **openChannel**, NO if not. An adaptor channel subclass should override this method without invoking EOAdaptorChannel's implementation.

See also: – **closeChannel**

lockRowComparingAttributes:entity:qualifier:snapshot:

– (void)**lockRowComparingAttributes:**(NSArray *)*attributes*
entity:(EOEntity *)*entity*
qualifier:(EOQualifier *)*qualifier*
snapshot:(NSDictionary *)*snapshot*

Attempts to lock a row in the database by selecting it with locking on. The lock operation succeeds if a select statement generated with *qualifier* retrieves exactly one row and the values in the row match the values in *snapshot*.

lockRowComparingAttributes:entity:qualifier:snapshot: invokes **selectAttributes:fetchSpecification:lock:entity:** with *attributes* as the attributes to select, a fetch specification built from *qualifier*, locking on, and *entity* as the entity. If the select returns no rows or more than one row, the method raises an EOGeneralAdaptorException. It also raises an EOGeneralAdaptorException if the values in the returned row don't match the corresponding values in *snapshot*.

The Framework uses this method whenever it needs to lock a row. When the Framework invokes it, *qualifier* specifies the primary key of the row to be locked and attributes used for locking to be compared in the database server. If any of the values specified in *qualifier* are different from the values in the database row, the select operation will not retrieve or lock the row. When this happens, the row to be locked has been updated in the database since it was last retrieved, and it isn't safe to update it.

Some attributes (such as BLOB types) can't be compared in the database. *attributes* should specify any such attributes. (If the row doesn't contain any such attributes, *attributes* can be **nil**.) If *qualifier* generates a select statement that returns and locks a single row, this method performs an in-memory comparison between the value in the retrieved row and the value in *snapshot* for each attribute in *attributes*. Therefore, *snapshot* must contain an entry for each attribute in *attributes*. In addition, it must contain an entry for the row's primary key.

Note: *snapshot*'s values are the values last fetched from the database for the attributes.

A subclass of EOAdaptorChannel doesn't need to override this method.

openChannel

– (void)**openChannel**

Implemented by subclasses to put the channel and both its context and adaptor into a state where they are ready to perform database operations. Raises an exception if an error occurs. An adaptor channel subclass should override this method without invoking EOAdaptorChannel's implementation.

See also: – **isOpen**, – **closeChannel**

performAdaptorOperation:

– (void)**performAdaptorOperation:**(EOAdaptorOperation *)*adaptorOperation*

Performs *adaptorOperation* by invoking the adaptor channel method appropriate for performing the specified operation. For example, if the adaptor operator for *adaptorOperation* is EOAdaptorInsertOperator, this method invokes **insertRow:forEntity:** using information in *adaptorOperation* to supply the arguments. Raises an exception if an error occurs.

A subclass of EOAdaptorChannel doesn't need to override this method.

See also: – **performAdaptorOperations:**

performAdaptorOperations:

– (void)**performAdaptorOperations:**(NSArray *)*adaptorOperations*

Performs adaptor operations by invoking **performAdaptorOperation:** with each EOAdaptorOperation object in the array *adaptorOperations*. An adaptor channel subclass may be able to override this method to take advantage of database-specific batch processing capabilities. Invokes the delegate methods **adaptorChannel:willPerformOperations:** and **adaptorChannel:didPerformOperations:exception:**.

This message raises an exception if an error occurs. The exception's userInfo dictionary contains these keys:

- EOAdaptorOperationsKey

  Corresponds to the array of adaptor operations that's being executed.

- EOFailedAdaptorOperationKey

  Corresponds to the particular adaptor operation that failed.

- EOAdaptorFailureKey

  If present, offers additional information on the type of error that occurred. Currently, the only possible value for this key is EOAdaptorOptimisticLockingFailure, which indicates that an update or lock operation failed because the row found in the database did not match the snapshot taken when the row was last fetched into the application.

A subclass of EOAdaptorChannel doesn't need to override the **performAdaptorOperations:** method.

primaryKeyForNewRowWithEntity:

– (NSDictionary *)**primaryKeyForNewRowWithEntity:**(EOEntity *)*entity*

Overridden by subclasses to return a primary key for a new row in the database table that corresponds to *entity*. If information in *entity* specifies an adaptor-specific means to assign a new primary key (for example, a sequence name or stored procedure), then this method returns a new primary key. Otherwise, if the key is a simple integer, the method tries to fetch a new primary key from the database using an adaptor-specific scheme. Otherwise, returns **nil**. EOAdaptorChannel's implementation returns **nil**. See your adaptor channel's documentation for information on how it generates primary keys.

A subclass of EOAdaptorChannel must override this method.

returnValuesForLastStoredProcedureInvocation

– (NSDictionary *)**returnValuesForLastStoredProcedureInvocation**

Implemented by subclasses to return stored procedure parameter and return values. Used in conjunction with **executeStoredProcedure:withValues:**. The dictionary returned by this method has entries whose keys are stored procedure parameter names and whose values are the parameter values. The dictionary also contains a special entry for the stored procedures return value with the key "returnValue". Returns an empty dictionary for stored procedures that have void return types. Returns **nil** if the stored procedure has results to fetch. In this case, you must use **fetchRowWithZone:** until there are no more results to fetch before the return value will be available.

An adaptor channel subclass should override this method without invoking EOAdaptorChannel's implementation.

selectAttributes:fetchSpecification:lock:entity:

– (void)**selectAttributes:**(NSArray *)*attributes*
fetchSpecification:(EOFetchSpecification *)*fetchSpecification*
lock:(BOOL)*flag*
entity:(EOEntity *)*entity*

Implemented by subclasses to select *attributes* in rows matching the qualifier in *fetchSpecification* and set the receiver's attributes to fetch. The selected rows compose one or more result sets, each row of which will be returned by subsequent **fetchRowWithZone:** messages according to *fetchSpecification*'s sort orderings. If *flag* is YES, the rows are locked if possible so that no other user can modify them (the lock specification in *fetchSpecification* is ignored). Raises an exception if an error occurs. Some possible reasons for failure are:

- The adaptor channel is in an invalid state (for example, fetching).
- The database failed to lock the specified rows.

An adaptor channel subclass should override this method without invoking EOAdaptorChannel's implementation.

See also: – **setAttributesToFetch:**

setAttributesToFetch:

– (void)**setAttributesToFetch:**(NSArray *)*attributes*

Implemented by subclasses to change the set of attributes used to describe the fetch data in the middle of a select. This method is normally invoked after **evaluateExpression:** but before the first call to **fetchRowWithZone:**. This is especially useful for database systems that may return non-rectangular result sets (like Sybase). This method raises if invoked when there is no fetch in progress.

An adaptor channel subclass should override this method without invoking EOAdaptorChannel's implementation.

See also: – **attributesToFetch**, – **selectAttributes:fetchSpecification:lock:entity:**

setDebugEnabled:

– (void)**setDebugEnabled:**(BOOL)*flag*

Enables debugging in the receiver and all its channels. If *debugEnabled* is YES, enables debugging; otherwise, disables debugging. When debugging is enabled, the adaptor channel logs evaluated SQL and other useful debugging information to the console (or to the standard error stream). The information provided may vary from adaptor to adaptor and may change from release to release.

A subclass of EOAdaptorContext doesn't need to override this method. A subclass that does override it must incorporate the superclass's version through a message to **super**.

See also: – **isDebugEnabled**, – **setDebugEnabled:** (EOAdaptorContext)

setDelegate:

– (void)**setDelegate:**(id)*delegate*

Sets the adaptor channel's delegate to *delegate*. The receiver does not retain *delegate*. A subclass of EOAdaptorContext doesn't need to override this method. A subclass that does override it must incorporate the superclass's version through a message to **super**.

See also: – **delegate**

updateValues:inRowDescribedByQualifier:entity:

– (void)**updateValues:**(NSDictionary *)*values*
inRowDescribedByQualifier:(EOQualifier *)*qualifier*
entity:(EOEntity *)*entity*

Updates the row described by *qualifier.* Invokes **updateValues:inRowsDescribedByQualifier:entity:** and raises an exception unless exactly one row is updated.

A subclass of EOAdaptorContext doesn't need to override this method.

updateValues:inRowsDescribedByQualifier:entity:

– (unsigned)**updateValues:**(NSDictionary *)*values*
inRowsDescribedByQualifier:(EOQualifier *)*qualifier*
entity:(EOEntity *)*entity*

Implemented by subclasses to update the rows described by *qualifier* with the values in *values*. *values* is an NSDictionary whose keys are attribute names and whose values are the new values for those attributes (the dictionary need only contain entries for the attributes being changed). Returns the number of updated rows. Raises an exception if an error occurs. Some possible reasons for failure are:

- The user logged in to the database doesn't have permission to update.
- The adaptor channel is in an invalid state (for example, fetching).
- The new values fail to satisfy a constraint defined in the database server.

An adaptor channel subclass should override this method without invoking EOAdaptorChannel's implementation.

See also: – **updateValues:inRowDescribedByQualifier:entity:**

Methods Implemented By the Delegate

adaptorChannelDidChangeResultSet:

– (void)**adaptorChannelDidChangeResultSet:**(id)*channel*

Invoked from **fetchRowWithZone:** when a select operation resulted in multiple result sets. This method tells the delegate that the next invocation of **fetchRowWithZone:** will fetch from the next result set. This method is invoked when **fetchRowWithZone:** returns **nil** and there are still result sets left to fetch. The delegate can invoke **setAttributesToFetch:** to prepare for fetching the new rows.

adaptorChannel:didEvaluateExpression:

– (void)**adaptorChannel:**(id)*channel*
didEvaluateExpression:(EOSQLExpression *)*expression*

Invoked from **evaluateExpression:** to tell the delegate that a query language expression has been evaluated by the database server.

adaptorChannel:didExecuteStoredProcedure:withValues:

– (void)**adaptorChannel:**(id)*channel*
didExecuteStoredProcedure:(EOStoredProcedure *)*procedure*
withValues:(NSDictionary *)*values*

Invoked from **executeStoredProcedure:withValues:** after the procedure is executed successfully.

adaptorChannel:didFetchRow:

– (void)**adaptorChannel:**(id)*channel* **didFetchRow:**(NSMutableDictionary *)*row*

Invoked from **fetchRowWithZone:** after a row is fetched successfully. This method is not invoked if an exception occurs during the fetch or if the same returns **nil** because there are no more rows in the current result set. The delegate may modify *row*, which will be returned from **fetchRowWithZone:**.

adaptorChannelDidFinishFetching:

– (void)**adaptorChannelDidFinishFetching:**(id)*channel*

Invoked from **fetchRowWithZone:** to tell the delegate that fetching is finished for the current select operation. This method is invoked when a fetch ends in **fetchRowWithZone:** because there are no more result sets.

adaptorChannel:didPerformOperations:exception:

– (NSException *)**adaptorChannel:**(id)*channel*
didPerformOperations:(NSArray *)*operations*
exception:(NSException *)*exception*

Invoked from **performAdaptorOperations:**. *exception* is **nil** if no exception was raised while *operations* were performed. Otherwise, *exception* is the raised exception. The delegate can return the same or a different exception, which is re-raised by **performAdaptorOperations:**, or it can return **nil** to prevent the adaptor channel from raising an exception.

adaptorChannel:didSelectAttributes:fetchSpecification:lock:entity:

– (void)**adaptorChannel:**(id)*channel*
didSelectAttributes:(NSArray *)*attributes*
fetchSpecification:(EOFetchSpecification *)*fetchSpecification*
lock:(BOOL)*flag*
entity:(EOEntity *)*entity*

Invoked from **selectAttributes:fetchSpecification:lock:entity:** to tell the delegate that rows have been selected in the database server.

adaptorChannelShouldConstructStoredProcedureReturnValues:

– (NSDictionary *)**adaptorChannelShouldConstructStoredProcedureReturnValues:**(id)*channel*

Invoked from **returnValuesForLastStoredProcedureInvocation** to tell the delegate that channel is constructing return values for the last stored procedure evaluated. If the delegate returns a value other than **nil**, that value will be returned immediately from **returnValuesForLastStoredProcedureInvocation**.

adaptorChannel:shouldEvaluateExpression:

– (BOOL)**adaptorChannel:**(id)*channel*
shouldEvaluateExpression:(EOSQLExpression *)*expression*

Invoked from **evaluateExpression:** to tell the delegate that *channel* is sending an expression to the database server. The delegate returns YES to permit the adaptor channel to send *expression* to the server. If the delegate returns NO, the adaptor channel does not send the expression and returns immediately. When the delegate returns NO, the adaptor channel expects that the implementor of the delegate has done the work that **evaluateExpression:** would have done. The delegate can create a new EOSQLExpression and send the expression itself before returning NO.

adaptorChannel:shouldExecuteStoredProcedure:withValues:

– (NSDictionary *)**adaptorChannel:**(id)*channel*
shouldExecuteStoredProcedure:(EOStoredProcedure *)*procedure*
withValues:(NSDictionary *)*values*

Invoked from **executeStoredProcedure:withValues:** to tell the delegate that *channel* is executing a stored procedure. If the delegate returns a value other than **nil**, that value is used as the arguments to the stored procedure instead of *values*.

adaptorChannel:shouldReturnValuesForStoredProcedure:

– (NSDictionary *)**adaptorChannel:**(id)*channel*
shouldReturnValuesForStoredProcedure:(NSDictionary *)*returnValues*

Invoked from **returnValuesForLastStoredProcedureInvocation** to tell the delegate that *channel* is returning values for a stored procedure. If the delegate returns a value other than **nil**, that value is returned from **returnValuesForLastStoredProcedureInvocation** instead of *returnValues*.

adaptorChannel:shouldSelectAttributes:fetchSpecification:lock:entity:

– (BOOL)**adaptorChannel:**(id)*channel*
shouldSelectAttributes:(NSArray *)*attributes*
fetchSpecification:(EOFetchSpecification *)*fetchSpecification*
lock:(BOOL)*flag*
entity:(EOEntity *)*entity*

Invoked from **selectAttributes:fetchSpecification:lock:entity:** to ask the delegate whether a select operation should be performed. The delegate should not modify *fetchSpecification*. Instead, if the delegate wants to perform a different select it should invoke **selectAttributes:fetchSpecification:lock:entity:** itself with a new fetch specification, and return NO (indicating that the adaptor channel should not perform the select itself).

adaptorChannelWillFetchRow:

– (void)**adaptorChannelWillFetchRow:**(id)*channel*

Invoked from **fetchRowWithZone:** to tell the delegate that a single row will be fetched. The delegate can determine the attributes used by the fetch by sending **attributesToFetch** to *channel*, and can change the set of attributes to fetch by sending **setAttributesToFetch:** to *channel*. The adaptor channel performs the actual fetch.

adaptorChannel:willPerformOperations:

– (NSArray *)**adaptorChannel:**(id)*channel* **willPerformOperations:**(NSArray *)*operations*

Invoked from **performAdaptorOperations:** to tell the delegate that *channel* is performing the EOAdaptorOperations in *operations*. The delegate may return *operations* or a different NSArray for the adaptor channel to perform. If the delegate returns **nil**, the adaptor channel does not perform the operations and returns from the method immediately.