InformixEOAdaptor Framework

Objective–C API Reference

Apple Computer, Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

InformixEOAdaptor

Framework: System/Library/Frameworks/InformixEOAdaptor.framework

Header File Directories: System/Library/Frameworks/InformixEOAdaptor.framework/Headers

Introduction

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

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

Class	Description
InformixAdaptor	Represents a single connection to a Informix database server, and is responsible for keeping login and model information, performing Informix-specific formatting of SQL expressions, and reporting errors.
InformixChannel	Represents an independent communication channel to the database server its InformixAdaptor is connected to.
InformixContext	Represents a single transaction scope on the database server to which its adaptor object is connected.
InformixSQLExpression	Defines how to build SQL statements for InformixChannels.

The Connection Dictionary

The connection dictionary contains items needed to connect to an Informix server, such as the database name (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 Informix adaptor defines string constants for use as connection dictionary keys:

- `dbNameKey`
- `userNameKey`
- `passwordKey`

See the InformixAdaptor class specification for more information on the connection dictionary key constants.

Locking

All adaptors use the database server's native locking facilities to lock rows on the server. In the Informix adaptor locking is determined by the isolation level, which is implemented in InformixChannel. Locking occurs when:

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

Informix Data Type	Java Data Type
VARCHAR	NSString
NVARCHAR	NSString
DECIMAL	NSDecimalNumber
MONEY	NSDecimalNumber
BYTE	NSData
TEXT	NSString
DATE	NSCalendarDate
INTEGER	NSNumber
SMALLINT	NSNumber
NCHAR	NSString
CHAR	NSNumber
SERIAL	NSNumber
FLOAT	NSNumber
SMALLFLOAT	NSNumber
DATETIME YEAR TO SECOND	NSCalendarDate
INTERVAL	NSString

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. InformixAdaptor does not make use of the model argument if one is provided.

Prototype Attributes

The InformixEOAdaptor Framework provides the following set of prototype attributes:

Name	External Type	Value Class Name	Other Attributes
binaryID	BYTE	NSData	
city	VARCHAR	NSString	columnName = CITY width = 50
date	DATETIME YEAR TO SECOND	NSCalendarDate	columnName = “”
longText	TEXT	NSString	
money	INTEGER	NSDecimalNumber	columnName = “”
phoneNumber	VARCHAR	NSString	columnName = PHONE width = 20
rawImage	BYTE	NSData	columnName = RAW_IMAGE
state	VARCHAR	NSString	columnName = STATE width = 2
streetAddress	VARCHAR	NSString	columnName = STREET_ADDRESS width = 100

Name	External Type	Value Class Name	Other Attributes
tiffImage	BYTE	NSImage	adaptorValueConversionMethodName = TIFFRepresentation columnName = PHOTO valueFactoryMethodName = “imageWithData:”
uniqueID	INTEGER	NSNumber	columnName = “” valueType = i
zipCode	VARCHAR	NSString	columnName = ZIP width = 10

Generating Primary Keys

Each adaptor provides a database-specific implementation of the method `primaryKeyForNewRowWithEntity:` for generating primary keys. The InformixChannel's implementation uses a table named `eo_sequence_table` to keep track of the next available primary key value for a given table. The table contains a row for each table for which the adaptor provides primary key values. The statement used to create the `eo_sequence_table` is:

```
create table eo_sequence_table (
    table_name varchar(32,0),
    counter integer
)
```

InformixChannel uses a stored procedure named `eo_pk_for_table` to access and maintain the primary key counter in `eo_sequence_table`. The stored procedure is defined as follows:

```
create procedure
eo_pk_for_table (tname varchar(32))
returning int;
    define cntr int;

    update EO_SEQUENCE_TABLE
    set COUNTER = COUNTER + 1
    where TABLE_NAME = tname;
```

```
    select COUNTER into cntr
    from EO_SEQUENCE_TABLE
    where TABLE_NAME = tname;

    return cntr;
end procedure;
```

The stored procedure increments the counter in the `eo_sequence_table` row for the specified table, selects the counter value, and returns it. InformixChannel executes this `eo_pk_for_table` stored procedure from `primaryKeyForNewRowWithEntity:` and returns the stored procedure's return value.

To use InformixChannel'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 InformixAdaptor 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 methods to operate on bind variables:

- `bindVariableDictionaryForAttribute:value:`
- `mustUseBindVariableForAttribute:`
- `shouldUseBindVariableForAttribute:`

InformixAdaptor

Inherits from: EOAdaptor : NSObject

Declared in: InformixEOAdaptor/InformixAdaptor.h

Class Description

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

The InformixAdaptor doesn't support full outer joins.

Constants

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

Constant	Corresponding value in the connection dictionary
`dbNameKey`	The name of the database.
`userNameKey`	The name of the user to log in as.
`passwordKey`	The user's password.

It defines a string constant for use as a key in an exception's userInfo dictionary (see `raiseInformixError:`).

Constant	Corresponding value in an exception's userInfo dictionary
`InformixErrorKey`	The error code raised in the Informix server. The value is an NSNumber with the error code as its `long` value, typically a negative number.

InformixAdaptor also defines a string constant to identify the user defaults domain for the Informix adaptor.

Constant	Description
`EOF_INFORMIX_ADAPTOR`	The name of the user defaults domain for the Informix adaptor.

Method Types

Mapping external types to internal types

```
+ externalToInternalTypeMap
```

Getting information from the connection dictionary

```
- informixConnectionString
- informixDefaultForKey:
- informixPassword
- informixUserName
- connectionKeys
```

Error handling

```
- raiseInformixError:
```

Preparing to connect

```
- prepareEnvironmentForConnect
- resetEnvironmentAfterConnect
```

Callback methods

- informixContextDidDisconnect:
- informixContextWillConnect:

Getting adaptor-specific classes

- adaptorContextClass
- adaptorChannelClass
- defaultExpressionClass

Class Methods

externalToInternalTypeMap

```
+ (NSDictionary *)externalToInternalTypeMap
```

Returns the mapping between each predefined external (database) type known by the adaptor to a default internal type. For information on the mapping, see the section in the InformixEOAdaptor Framework introduction titled "Data Type Mapping" (page 5).

Instance Methods

adaptorChannelClass

```
- (Class)adaptorChannelClass
```

Returns the InformixChannel class.

adaptorContextClass

```
- (Class)adaptorContextClass
```

Returns the InformixContext class.

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.

defaultExpressionClass

```
- (Class)defaultExpressionClass
```

Returns the InformixSQLExpression class.

informixConnectionString

```
- (NSString *)informixConnectionString
```

Returns the user name, password, and database name as a string suitable to be supplied as an argument to db_connect().

informixContextDidDisconnect:

```
- (void)informixContextDidDisconnect:(InformixContext *)logon
```

Callback method that is invoked after the associated Informix context disconnects.

informixContextWillConnect:

```
- (void)informixContextWillConnect:(InformixContext *)logon
```

Callback method that is invoked just before the associated Informix context disconnects.

informixDefaultForKey:

```
- (NSString *)informixDefaultForKey:(NSString *)key
```

Returns the user default setting for *key*. To get this information it first checks the user defaults, and then the adaptor's internal defaults dictionary.

informixPassword

```
- (NSString *)informixPassword
```

Returns the password in the connection dictionary.

informixUserName

```
- (NSString *)informixUserName
```

Returns the user name in the connection dictionary.

prepareEnvironmentForConnect

```
- (void)prepareEnvironmentForConnect
```

Prepares the user environment for connection to an Informix server. Preserves existing environment variables, replacing them with values obtained from the adaptor's connection dictionary. Unset variables are set to values obtained from `informixDefaultForKey:`, if any.

See Also: `- resetEnvironmentAfterConnect`

raiseInformixError:

```
- (void)raiseInformixError:(NSString *)sqlString
```

Examines Informix structures for error flags and raises an exception if one is found. Extracts the error information in the connection structure and uses it to build and raise an exception. The error code is available in the exception's user info dictionary in the entry for the key, `InformixErrorKey`.

resetEnvironmentAfterConnect

```
- (void)resetEnvironmentAfterConnect
```

Restores any environment variables overwritten by `prepareEnvironmentForConnect`.

InformixChannel

Inherits from: EOAdaptorChannel : NSObject

Declared in: InformixEOAdaptor/InformixChannel.h
InformixEOAdaptor/InformixDescription.h

Class Description

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

The features InformixChannel adds to EOAdaptorChannel are as follows:

- Informix-specific error handling (see InformixChannel Delegate)
- The ability to configure the fetch buffer
- The ability to read a list of table names from the database

Method Types

Finding table names

- `setInformixTableNamesSQL:`
- `informixTableNamesSQL`

Getting the cursor data area

- `cursorDataArea`

Setting the isolation level

- `informixSetIsolationTo:`

Setting the fetch buffer length

- `setFetchBufferLength:`
- `fetchBufferLength`

Instance Methods

cursorDataArea

```
- (struct informix_cursor *)cursorDataArea
```

If the channel is connected, returns an Informix-specific data structure describing characteristics of the channel. Otherwise, returns NULL.

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.

informixSetIsolationTo:

```
- (void)informixSetIsolationTo:(InformixIsolationLevel)isolationLevel
```

Sets the isolation transaction level of the connection represented by the receiver to *isolationLevel*.

informixTableNamesSQL

```
- (NSString *)informixTableNamesSQL
```

Returns the SQL statement the receiver uses to find table names. The user default InformixTableNamesSQL overrides a statement set with `setInformixTableNamesSQL:`.

setFetchBufferLength:

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

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

setInformixTableNamesSQL:

```
- (void)setInformixTableNamesSQL:(NSString *)sql
```

Set the SQL statement the receiver uses to find table names to *sql*.

InformixContext

Inherits from: EOAdaptorContext : NSObject

Declared in: InformixEOAdaptor/InformixContext.h

Class Description

An InformixContext 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 InformixContext may in turn have several InformixChannels, which handle actual access to the data on the server.

The features the InformixContext class adds to EOAdaptorContext are methods for setting Informix-specific characteristics for the context.

Method Types

Managing a connection to the server

- `connect`
- `connection`
- `disconnect`

- isConnected

Returning information about an InformixContext

- fetchesInProgress
- hasTransactions

Returns information about the server

- isOnLine

Instance Methods

connect

```
- (void)connect
```

Opens a connection to the database server. An InformixChannel sends this message to its InformixContext when the channel is about to open a connection to the server.

See Also: - disconnect

connection

```
- (long)connection
```

Returns an identifier for the receiver's connection to the server.

disconnect

```
- (void)disconnect
```

Closes a connection to the database server. An InformixChannel sends this message to its InformixContext when the channel is about to close a connection to the server.

See Also: - connect

fetchesInProgress

```
- (unsigned)fetchesInProgress
```

Returns the number of fetches the receiver has in progress.

hasTransactions

```
- (BOOL)hasTransactions
```

Returns `YES` to indicate that the receiver has transactions in process, `NO` otherwise.

isConnected

```
- (BOOL)isConnected
```

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

See Also: `- connect`, `- disconnect`

isOnLine

```
- (BOOL)isOnLine
```

Returns `YES` if the server is an Informix online server, `NO` otherwise.

InformixSQLExpression

Inherits from: EOSQLExpression : NSObject

Declared in: InformixEOAdaptor/InformixSQLExpression.h

Class Description

InformixSQLExpression defines how to build SQL statements for InformixChannels.

Bind Variables

The InformixAdaptor 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 methods to operate on bind variables:

- `bindVariableDictionaryForAttribute:value:`
- `mustUseBindVariableForAttribute:`
- `shouldUseBindVariableForAttribute:`

For more information on using bind variables, see the EOSQLExpression class specification.

Class Methods

serverTypeIdForName

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

Returns the Informix type code (such as InfDecimal, InfDate, or InfCHAR) for *typeName* (such as "DECIMAL", "DATE", or "CHAR").

Instance Methods

lockClause

```
- (NSString *)lockClause
```

Overrides the EOSQLExpression method `lockClause` to return the SQL string used in a SELECT statement to lock selected rows, which is @"FOR UPDATE OF".

InformixChannel Delegate

(informal protocol)

Declared in: InformixEOAdaptor/InformixChannel.h

Protocol Description

InformixChannel's delegate allows you to process errors that occur in the Informix server.

Instance Methods

informixChannel:willReportDatabaseError:

```
- (BOOL)informixChannel:(InformixChannel *)channel
   willReportDatabaseError:(NSString *)error
```

Invoked whenever *channel* encounters an error reported by the Informix server. The *error* argument is the text of the Informix error message which will be sent to the adaptor's `reportError:` method. The delegate can return `NO` to prevent the channel from calling `reportError:`.

This Apple manual was written, edited, and composed on a desktop publishing system using Apple Macintosh computers and FrameMaker software.

Line art was created using Adobe™ Illustrator and Adobe Photoshop.

Text type is Palatino® and display type is Helvetica®. Bullets are ITC Zapf Dingbats®. Some elements, such as program listings, are set in Adobe Letter Gothic.