

The InformixEOAdaptor Framework

Framework: com.apple.yellow.informixeadaptor

Header File Directories: System/Developer/Java/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. For Informix databases the required keys are as follows:

- dbName
- userName
- password

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 **selectAttributesWithFetchSpecification** message with true specified as the value for the **lock** parameter.
- You explicitly lock an object's row with the EODatabaseContext's **lockObjectWithGlobalID** 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	Objective-C Data Type	Java Data Type
VARCHAR	NSString	String
NVARCHAR	NSString	String
DECIMAL	NSDecimalNumber	BigDecimal
MONEY	NSDecimalNumber	BigDecimal
BYTE	NSData	NSData
TEXT	NSString	String
DATE	NSDate	NSDate
INTEGER	NSNumber	Number
SMALLINT	NSNumber	Number
NCHAR	NSString	String
CHAR	NSString	Number
SERIAL	NSNumber	Number
FLOAT	NSNumber	Number
SMALLFLOAT	NSNumber	Number

Informix Data Type	Objective-C Data Type	Java Data Type
DATETIME YEAR TO SECOND	NSDate	NSGregorianCalendar
INTERVAL	NSString	String

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"	NSDate	columnName = ""
longText	TEXT	NSString	
money	INTEGER	NSNumber	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;
tiffImage	BYTE	UIImage	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 **primaryKeyForNewRow** 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 **primaryKeyForNewRow** 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:

- `bindValueDictionaryForAttribute`

- `mustUseBindVariableForAttribute`
- `shouldUseBindVariableForAttribute`

InformixAdaptor

Inherits From: EOAdaptor : NSObject
Package: com.apple.yellow.informixeadaptor

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

Method Types

Working with channels and contexts	adaptorChannelClass adaptorContextClass
Getting information from the connection dictionary	informixConnectionString informixDefaultForKey connectionKeys
Error handling	raiseInformixError

Instance Methods

adaptorChannelClass

```
public java.lang.Class adaptorChannelClass()
```

Returns the InformixChannel class.

adaptorContextClass

```
public java.lang.Class adaptorContextClass()
```

Returns the InformixContext class.

connectionKeys

```
public com.apple.yellow.foundation.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.

informixConnectionString

```
public java.lang.String informixConnectionString()
```

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

informixDefaultForKey

```
public java.lang.String informixDefaultForKey(java.lang.String aString)
```

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.

raiseInformixError

```
public void raiseInformixError(java.lang.String aString)
```

Examines Informix structures for error flags and raises an exception if one is found. Extracts the error information in the connection structure and use it to build and raise an exception.

InformixChannel

Inherits From: EOAdaptorChannel : NSObject

Package: com.apple.yellow.informixeadaptor

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
- The ability to configure the fetch buffer
- The ability to read a list of table names from the database

Method Types

Setting the fetch buffer length	setFetchBufferLength
	fetchBufferLength

Instance Methods

fetchBufferLength

```
public int 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.

See also: [setFetchBufferLength](#)

setFetchBufferLength

public void **setFetchBufferLength**(int *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.

See also: **fetchBufferLength**

InformixContext

Inherits From: EOAdaptorContext : NSObject

Package: com.apple.yellow.informixeadaptor

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
- isOnLine

Instance Methods

connect

```
public void connect()
```

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

See also: **disconnect**

connection

public int **connection**()

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

disconnect

public void **disconnect**()

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

See also: **connect**

fetchesInProgress

public int **fetchesInProgress**()

Returns the number of fetches the receiver has in progress.

hasTransactions

public boolean **hasTransactions**()

Returns **true** to indicate that the receiver has transactions in process, **false** otherwise.

isConnected

public boolean **isConnected**()

Returns **true** if the receiver has an open connection to the database, **false** otherwise.

See also: **connect, disconnect, isConnected**

isOnLine

public boolean **isOnLine**()

Returns **true** if Is the server an Informix on-line server, **false** otherwise.

InformixSQLExpression

Inherits From: EOSQLExpression : NSObject

Package: com.apple.yellow.informixeadaptor

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:

- `bindValueDictionaryForAttribute`
- `mustUseBindVariableForAttribute`
- `shouldUseBindVariableForAttribute`

Static Methods

formatValueForAttribute

```
public static java.lang.String formatValueForAttribute(java.lang.Object value,  
com.apple.yellow.eoaccess.EOAttribute attribute)
```

Overrides the EOSQLExpression method **formatValueForAttribute** to return a formatted string representation of *value* for *attribute* that is suitable for use in a SQL statement.

serverTypeIdForName

```
public static int serverTypeIdForName(java.lang.String typeName)
```

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

Instance Methods

bindValueDictionaryForAttribute

```
public com.apple.yellow.foundation.NSMutableDictionary  
    bindValueDictionaryForAttribute(com.apple.yellow.eoaccess.EOAttribute attribute,  
    java.lang.Object value)
```

Overrides the EOSQLExpression method **bindValueDictionaryForAttribute** to return the receiver's bind variable dictionaries. For more information on bind variables, see the discussion in the class description.

See also: **mustUseBindVariableForAttribute**, **shouldUseBindVariableForAttribute**

mustUseBindVariableForAttribute

```
public boolean  
    mustUseBindVariableForAttribute(com.apple.yellow.eoaccess.EOAttribute attribute)
```

Overrides the EOSQLExpression method **mustUseBindVariableForAttribute** to return YES if the receiver must use bind variables for *attribute*, NO otherwise. A returned value of YES indicates that the underlying RDBMS requires that bind variables be used for attributes with *attribute*'s external type.

See also: **bindValueDictionaryForAttribute**, **shouldUseBindVariableForAttribute**

shouldUseBindVariableForAttribute

```
public boolean  
    shouldUseBindVariableForAttribute(com.apple.yellow.eoaccess.EOAttribute attribute)
```

Overrides the EOSQLExpression method **shouldUseBindVariableForAttribute** to return YES if the receiver can provide a bind variable dictionary for *attribute*, NO otherwise. A returned value of YES indicates that the receiver should use bind variables for attributes with *attribute*'s external type.

See also: **bindValueDictionaryForAttribute**, **mustUseBindVariableForAttribute**