The InformixEOAdaptor Framework

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. 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 **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	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	NSCalendarDate	NSGregorianDate
INTEGER	NSNumber	Number
SMALLINT	NSNumber	Number
NCHAR	NSString	String
CHAR	NSString	Number
SERIAL	NSNumber	Number
FLOAT	NSNumber	Number

Informix Data Type	Objective-C Data Type	Java Data Type
SMALLFLOAT	NSNumber	Number
DATETIME YEAR TO SECOND	NSCalendarDate	NSGregorianDate
INTERVAL	NSString	String

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

Method Types

Mapping external types to internal types	externalTypesWithModel: internalTypeForExternalType:model:
Working with channels and contexts	adaptorChannelClass adaptorContextClass
Testing the connection dictionary	assertConnectionDictionaryIsValid
Getting information from the connection dictionary	informixConnectionString informixDefaultForKey: connectionKeys
Getting the default expression class	defaultExpressionClass
Verifying a qualifier type	isValidQualifierType:model:
Error handling	raiseInformixError:

Class Methods

externalTypesWithModel:

+ (NSArray *)**externalTypesWithModel:**(EOModel *)*model*

Overrides the EOAdaptor method **externalTypesWithModel:** to return the Informix database types.

See also: **internalTypeForExternalType:model:**

internalTypeForExternalType:model:

+ (NSString *)**internalTypeForExternalType:**(NSString *)*externalType* **model:**(EOModel *)*model*

Overrides the EOAdaptor method **internalTypeForExternalType:model:** to return the name of the Objective-C class used to represent values stored in the database as *externalType*.

See also: **externalTypesWithModel:**

Instance Methods

adaptorChannelClass

– (Class)**adaptorChannelClass**

Returns the InformixChannel class.

adaptorContextClass

– (Class)**adaptorContextClass**

Returns the InformixContext class.

assertConnectionDictionaryIsValid

– (void)**assertConnectionDictionaryIsValid**

Overrides the EOAdaptor method **assertConnectionDictionaryIsValid** to verify that the receiver can connect to the database with its connection dictionary. Briefly forms a connection to the server to validate the connection dictionary and then closes the connection (in other words, this method doesn't open a connecton to the database—that happens when the first adaptor channel is sent an **openChannel** message). The adaptor uses this method in conjunction with displaying a server login panel. Raises an exception if an error occurs.

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().

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.

isValidQualifierType:model:

– (BOOL)**isValidQualifierType:**(NSString *)*typeName* **model:**(EOModel *)*model*

Overrides the EOAdaptor method **isValidQualifierType:model:** to return YES if an attribute of type *typeName* can be used in a qualifier (a SQL WHERE clause) sent to the database server, or NO otherwise. *typeName* is the name of a type as required by the database server, such as an Informix "VARCHAR".

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 use it to build and raise an exception.

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

Method Types

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.

See also: – **setFetchBufferLength:**

informixSetIsolationTo:

– (void)**informixSetIsolationTo:**(InformixIsolationLevel)*isol*

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

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.

See also: – **fetchBufferLength**

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
	– isOnLine

Instance Methods

connect

– (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

– (long)**connection**

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

disconnect

– (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

– (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**, – **isConnected**

isOnLine

– (BOOL)**isOnLine**

Returns YES if Is the server an Informix on-line 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:

Class Methods

formatValue:forAttribute:

+ (NSString *)**formatValue:**(id)*value* **forAttribute:**(EOAttribute *)*attribute*

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

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

bindVariableDictionaryForAttribute:value:

- (NSMutableDictionary *)**bindVariableDictionaryForAttribute:**(EOAttribute *)*attribute* **value:***value*

Overrides the EOSQLExpression method **bindVariableDictionaryForAttribute:value:** 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:

- (BOOL)**mustUseBindVariableForAttribute:**(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: – **bindVariableDictionaryForAttribute:value:**, – **shouldUseBindVariableForAttribute:**

shouldUseBindVariableForAttribute:

- (BOOL)**shouldUseBindVariableForAttribute:**(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: – **bindVariableDictionaryForAttribute:value:**, – **mustUseBindVariableForAttribute:**