



# SybaseEOAdaptor Framework

Java API Reference



Apple Computer, Inc.  
© 1999 Apple Computer, Inc.  
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Computer, Inc., except to make a backup copy of any documentation provided on CD-ROM.

The Apple logo is a trademark of Apple Computer, Inc. Use of the “keyboard” Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this book. Apple retains all intellectual property rights associated with the technology described in this book. This book is intended to assist application developers to develop applications only for Apple-labeled or Apple-licensed computers.

Every effort has been made to ensure that the information in this manual is accurate. Apple is not responsible for typographical errors.

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

Apple, the Apple logo, Macintosh, and WebObjects are trademarks of Apple Computer, Inc., registered in the United States and other countries. Enterprise Objects is a trademark of Apple Computer, Inc.

NeXT, the NeXT logo, OPENSTEP, Enterprise Objects Framework, Objective-C, and WEBSOCKET are trademarks of NeXT Software, Inc.

Adobe, Acrobat, and PostScript are trademarks of Adobe Systems Incorporated or its subsidiaries and may be registered in certain jurisdictions.

Helvetica and Palatino are registered trademarks of Linotype-Hell AG and/or its subsidiaries.

ITC Zapf Dingbats is a registered trademark of International Typeface Corporation.

ORACLE is a registered trademark of Oracle Corporation, Inc.

SYBASE is a registered trademark of Sybase, Inc.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

Windows NT is a trademark of Microsoft Corporation.

All other trademarks mentioned belong to their respective owners.

Simultaneously published in the United States and Canada.

**Even though Apple has reviewed this manual, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS MANUAL, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS MANUAL IS SOLD “AS IS,” AND YOU, THE PURCHASER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.**

**IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS MANUAL, even if advised of the possibility of such damages.**

**THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.**

**Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.**

# SybaseEOAdaptor Framework

---

**Package:** com.apple.yellow.sybaseeoadaptor

## Introduction

---

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

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

<b>Class</b>	<b>Description</b>
SybaseAdaptor	Represents a single connection to a Sybase database server, and is responsible for keeping login and model information, performing Sybase-specific formatting of SQL expressions, and reporting errors.
SybaseChannel	Represents an independent communication channel to the database server its SybaseAdaptor is connected to.
SybaseContext	Represents a single transaction scope on the database server to which its adaptor object is connected.
SybaseSQLExpression	Defines how to build SQL statements for SybaseChannels.

## The Connection Dictionary

---

The connection dictionary contains items needed to connect to a Sybase 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 Sybase adaptor defines string constants for use as connection dictionary keys:

- HOSTNAME
- DATABASENAME
- USERNAME
- PASSWORD
- LC\_ALL\_KEY
- ENCRYPTPASSWORD
- PRIMITIVE\_TYPE\_MAP

The last three keys are optional. The `ENCRYPTPASSWORD` key provides support for Sybase password encryption. A value for the `LC_ALL_KEY` declares to the Sybase server the character set being used by the client (such as `euclj`, `ascii7`, or `iso_1`). For a complete list of types available for this field, see your Sybase documentation. The `PRIMITIVE_TYPE_MAP` entry describes the mapping of user-defined data types to their base Sybase type (such as `varchar` or `datetime`). For more information on user-defined data types, see [“Data Type Mapping”](#) (page 6).

To add any of these optional keys and appropriate values to your connection dictionary, you can manually edit your model file. For example:

```
connectionDictionary = {databaseName = People;
    hostName = "";
    LC_ALL = euclj;
    password = "";
    primitiveTypeMap = {id = varchar; ssn = char(9); };
    sybasePasswordEncryption = YES;
    userName = "";
};
```

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. Alternatively you can add the optional keys to a model's connection dictionary programmatically.

The default character set for non-Japanese systems is `iso_1` (that is, ISO Latin 1), while the default character set for Japanese systems is `eucljis`. You only need to add the `LC_ALL` key to your connection dictionary if you are using a character set other than your system's default.

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

## Error Handling

---

`SybaseAdaptor`, `SybaseContext`, and `SybaseChannel` can raise exceptions due to programming errors that result in invalid argument values or internal inconsistencies. In addition, messages, errors, and failure status returned from the Sybase SQL Server and client libraries can also result in `EOGeneralAdaptorExceptions`. When an exception results from a callback to the `CS_CLIENTMSG_CB` (Sybase ClientMessage callback) or the `CS_SERVERMSG_CB` (Sybase ServerMessage callback), all of the information passed into this routine is available in the `userInfo` dictionary contained by the exception. When an exception is raised in response to a Sybase ClientMessage callback, you can get the information provided by the client library as follows:

```
clientMsgDict =  
    localException().userInfo().objectForKey("sybaseClientMessageDictionary");
```

The `clientMsgDict` contains the following keys which have values corresponding to those sent in the callback function that raised the exception: `msgstring`, `osstring`, `sqlstate`, `severity`, `msgnumber`, `osnumber`, `status`.

Similarly, when the exception is raised in response to a Sybase ServerMessage callback, you can get the information provided by the server as follows:

```
svrMsgDict = localException().userInfo().objectForKey("sybaseServerMessageDictionary");
```

The `svrMsgDict` contains the following keys which have values corresponding to those sent in the callback function that raised the exception: `text`, `svrname`, `proc`, `sqlstate`, `msgnumber`, `state`, `severity`, `line`, `status`.

## Locking

---

All adaptors use the database server's native locking facilities to lock rows on the server. The Sybase adaptor locks a row by using the HOLDLOCK keyword in SELECT statements. This occurs when:

- You send the adaptor channel a `selectAttributes` message with `true` specified as the value for the lock keyword.
- You explicitly lock an object's row with the `EODatabaseContext`'s `lockObjectWithGlobalID` message.
- You set pessimistic locking at the database level and fetch objects.

The semantics of the HOLDLOCK keyword are such that when you lock a row other users can't update it, but it doesn't guarantee that your update will succeed. This is because other users could be holding a lock on the same row. However, you can still read rows that are locked by other users.

## Data Type Mapping

---

Every adaptor provides a mapping between each server data type (the **external** type) and the Java type (the **internal** type) to which a database value will be coerced when it's fetched from the database. The following table lists the mapping used by SybaseAdaptor.

External Data Type	Internal Data Type
binary	NSData
bit	Number
char	String
datetime	NSDate
datetimn	NSDate
decimal	BigDecimal
decimaln	BigDecimal
float	Number
floatn	Number

External Data Type	Internal Data Type
image	NSData
int	Number
intn	Number
money	BigDecimal
moneyn	BigDecimal
nchar	String
numeric	BigDecimal
numericn	BigDecimal
nvarchar	String
real	Number
smalldatetime	NSGregorianCalendar
smallint	Number
smallmoney	BigDecimal
sysname	String
text	String
timestamp	NSData
tinyint	Number
varbinary	NSData
varchar	String

In addition, SybaseAdaptor provides a mapping for user-defined data types. For example, a custom data type `partnumber` defined as `char(10)` is mapped to `String`—the Java type to which `partnumber`'s base data type (`char`) is mapped. SybaseAdaptor's implementation of `describeModelWithTableNames` automatically creates mappings for user-defined data types and saves them in the connection dictionary of the newly created model. Consequently, even models created with EOModeler automatically include information about custom data types.

Since information about custom types is stored in a model's connection dictionary, the type mapping methods—`externalToInternalTypeMap`, `internalTypeForExternalType`, and `isValidQualifierType`—use the model argument if it is provided. If the model argument isn't provided, these methods don't have user-defined data type information available to them.

## Prototype Attributes

The SybaseEOAdaptor Framework provides the following set of prototype attributes:

<b>Name</b>	<b>External Type</b>	<b>Value Class Name</b>	<b>Other Attributes</b>
binaryID	varbinary	NSData	width = 12
city	varchar	String	columnName = CITY width = 50
date	datetime	NSGregorianCalendar	columnName = ""
longText	text	String	
money	money	BigNumber	columnName = ""
phoneNumber	varchar	String	columnName = PHONE width = 20
rawImage	image	NSData	columnName = RAW_IMAGE
state	varchar	String	columnName = STATE width = 2
streetAddress	varchar	String	columnName = STREET_ADDRESS width = 100

Name	External Type	Value Class Name	Other Attributes
tiffImage	image	NSImage	adaptorValueConversionMethodName = TIFFRepresentation columnName = PHOTO valueFactoryMethodName = "imageWithData:"
uniqueID	int	Number	columnName = "" valueType = i
zipCode	varchar	String	columnName = ZIP width = 10

## Generating Primary Keys

Each adaptor provides a database-specific implementation of the method `primaryKeyForNewRowWithEntity` for generating primary keys. The `SybaseChannel`'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),
    counter int null
)
```

`SybaseChannel` 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) as
begin
    define @max int

    update eo_sequence_table
    set counter = counter + 1
    where table_name = @tname
```

## FRAMEWORK SybaseEOAdaptor Framework

```
select counter
from eo_sequence_table
where table_name = @tname
end
```

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

To use `SybaseChannel`'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*.

# SybaseAdaptor

---

**Inherits from:** EOAdaptor : NSObject  
**Package:** com.apple.yellow.sybaseeoadaptor

## Class Description

---

A SybaseAdaptor represents a single connection to a Sybase database server, and is responsible for keeping login and model information, performing Sybase-specific formatting of SQL expressions, and reporting errors.

The features SybaseAdaptor adds to EOAdaptor are as follows:

- The ability to specify a client character set and language
- Sybase password encryption

The SybaseAdaptor class has these restrictions: A context can only manage one channel at a time, and the adaptor doesn't support full outer joins because the Sybase server itself doesn't support them.

## Constants

---

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

Constant	Corresponding value in the connection dictionary
HOSTNAME	The name of the machine on which the database server runs.
DATABASENAME	The name of the database.
USERNAME	The name of the user to log in as.
PASSWORD	The user's password.
LC_ALL_KEY	The setting to LC_ALL, which is used to specify the language and character set for server connections. On J systems this option defaults to japanese.
ENCRYPTPASSWORD	Either the string "Yes" or the string "No". If the value is "Yes", the adaptor enables the System 10 password encryption feature before attempting to open a connection.
PRIMITIVE_TYPE_MAP	A dictionary representing the database's primitive type map. Sybase allows user defined types in a database that map onto primitive types. For instance, you could define a user type, <code>primary_key</code> , and map it to <code>int</code> . The primitive type map dictionary would then contain the key "primary_key" with the value "int".

For more information on the connection dictionary, see ["The Connection Dictionary"](#) (page 4) in the SybaseEOAdaptor framework introduction.

## Method Types

---

### Mapping external types to internal types

- `externalToInternalTypeMap`
- `primitiveTypeForExternalTypeInModel`

### Getting information from the connection dictionary

- `connectionKeys`

### Bracketing calls to `ct_connect()`

- `prepareEnvironmentForConnect`
- `resetEnvironmentAfterConnect`

### Callback methods

- `sybaseContextDidDisconnect`
- `sybaseContextWillConnect`

## Static Methods

---

### **externalToInternalTypeMap**

```
public static 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 SybaseEOAdaptorFramework introduction titled [“Data Type Mapping”](#) (page 6).

## CLASS SybaseAdaptor

### primitiveTypeForExternalTypeInModel

```
public static String primitiveTypeForExternalTypeInModel(  
    String externalType,  
    com.apple.yellow.eoaccess.EOModel model)
```

Returns the primitive type on which a given custom type, defined on the server, is based.

## Instance Methods

---

### connectionKeys

```
public 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.

### prepareEnvironmentForConnect

```
public void prepareEnvironmentForConnect()
```

A call to this method should precede all calls to `ct_connect()` to set the `LC_ALL` environment variable setting to the value specified in the model connection dictionary.

**See Also:** `resetEnvironmentAfterConnect`

### resetEnvironmentAfterConnect

```
public void resetEnvironmentAfterConnect()
```

A call to this method should follow all calls to `ct_connect()` to set the `LC_ALL` environment variable setting to the value specified in the model connection dictionary.

**See Also:** `prepareEnvironmentForConnect`

## **CLASS SybaseAdaptor**

### **sybaseContextDidDisconnect**

```
public void sybaseContextDidDisconnect(SybaseContext aSybaseContext)
```

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

### **sybaseContextWillConnect**

```
public void sybaseContextWillDisconnect(SybaseContext aSybaseContext)
```

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

**CLASS SybaseAdaptor**

# SybaseChannel

---

**Inherits from:** EOAdaptorChannel : NSObject

**Package:** com.apple.yellow.sybaseeoadaptor

## Class Description

---

A SybaseChannel represents an independent communication channel to the database server its SybaseAdaptor is connected to. All of a SybaseChannel's operations take place within the context of transactions controlled or tracked by its SybaseContext. A Sybase adaptor context manages one channel, and a channel is associated with only one context.

**CLASS SybaseChannel**

# SybaseContext

---

**Inherits from:** EOAdaptorContext : NSObject

**Package:** com.apple.yellow.sybaseeoadaptor

## Class Description

---

A SybaseContext represents a single transaction scope on the database server to which its adaptor object is connected. Since a Sybase server supports multiple concurrent transaction sessions, the adaptor may have several adaptor contexts. A SybaseContext may in turn have a SybaseChannel, which handles actual access to the data on the server.

The SybaseContext can have a delegate, which gives you access to all messages returned from the Sybase client library or from the Sybase Server. See the SybaseContext.Delegate interface specification for a complete description.

## Method Types

---

Setting the login time out interval

```
loginTimeInterval  
setLoginTimeInterval
```

## CLASS SybaseContext

### Setting the time out interval

`setTimeoutInterval`

`timeoutInterval`

### Managing the connection

`connect`

`currentChannel`

`disconnect`

`isConnected`

`setMaximumConnections`

`maximumConnections`

### Setting the max text size default

`maxTextSizeDefault`

`setMaxTextSizeDefault`

### Setting the current exception

`raiseCurrentException`

`setCurrentException`

## Static Methods

---

### **maximumConnections**

```
public static int maximumConnections()
```

Returns the value for CS\_MAX\_CONNECT, the maximum number of database connections a Sybase client process can have open simultaneously.

## CLASS SybaseContext

### loginTimeoutInterval

```
public static int loginTimeoutInterval()
```

Returns the login time out interval used by SybaseContext.

**See Also:** `setLoginTimeoutInterval`

### setLoginTimeoutInterval

```
public static void setLoginTimeoutInterval(int seconds)
```

Sets the login time out interval value SybaseContext uses during the creation of new channels. The default is 0, which means that there is no time out.

**See Also:** `loginTimeoutInterval`

### setMaximumConnections

```
public static boolean setMaximumConnections(int value)
```

Sets CS\_MAX\_CONNECT, the maximum number of database connections a Sybase client process can have open simultaneously. Returns `true` if the operation is successful. By default the adaptor uses the Sybase client library default, which is normally sufficient. However, if your application communicates with many databases or uses database authentication for each user, you might need to raise the limit. It is possible to raise the limit after database connections have been opened.

### setTimeoutInterval

```
public static void setTimeoutInterval(int seconds)
```

Sets the time out interval value that the SybaseContext uses during the creation of new channels. The default is 0, which means that there is no time out.

**See Also:** `timeoutInterval`

## CLASS SybaseContext

### timeOutInterval

```
public static int timeOutInterval()
```

Returns the time out interval used by SybaseContext.

## Instance Methods

---

### connect

```
public void connect()
```

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

**See Also:** disconnect

### currentChannel

```
public SybaseChannel currentChannel()
```

Returns the SybaseChannel currently associated with the receiving context.

### disconnect

```
public void disconnect()
```

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

**See Also:** connect

## CLASS SybaseContext

### isConnected

```
public boolean isConnected()
```

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

**See Also:** `connect`, `disconnect`

### maxTextSizeDefault

```
public int maxTextSizeDefault()
```

Returns the maximum number of bytes to be returned from a Sybase image to text field. The default is set to `INT_MAX`, as defined for the host machine. This number can be overwritten on a per-channel basis by sending the appropriate SQL to the channel using the `evaluateExpression:` method.

### raiseCurrentException

```
public void raiseCurrentException()
```

If the receiver has an exception, raises it.

**See Also:** `setCurrentException`

### setCurrentException

```
public void setCurrentException(Throwable exception)
```

Sets the receiver's current exception to *exception*.

When the SybaseAdaptor encounters an error, it uses the error message to build an `NSEException` and stores the exception in the `SybaseContext` using this method. The exception can then be reviewed by other components to determine if the error is fatal.

**See Also:** `raiseCurrentException`

## CLASS `SybaseContext`

### **`setMaxTextSizeDefault`**

```
public void setMaxTextSizeDefault(int textSize)
```

Sets the receiver's default text size to *textSize*. Any channels created after this method has been invoked will use the newly specified text size.

# SybaseSQLExpression

---

**Inherits from:** EOSQLExpression : NSObject

**Package:** com.apple.yellow.sybaseoadaptor

## Class Description

---

SybaseSQLExpression defines how to build SQL statements for SybaseChannels.

## Static Methods

---

### **serverTypeIdForName**

```
public static int serverTypeIdForName(String typeName)
```

Returns the Sybase type code (such as 47, 56, or 55) for *typeName* (such as "char", "int", or "decimal").

## Instance Methods

---

### **lockClause**

```
public String lockClause()
```

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

# SybaseContext.Delegate

---

(informal interface)

**Package:** com.apple.yellow.sybaseeoadaptor

## Interface Description

---

The SybaseContext delegate object allows developers access to all the messages returned from the Sybase client library or the Sybase Server. If your implementation of these delegate methods returns `false`, the SybaseContext will not report the message (or error). If your implementation returns `true`, the SybaseContext will continue as usual. Most messages are reported in exceptions, but messages with a severity of 0 are simply ignored.

## Instance Methods

---

### **sybaseContextShouldReportClientMessage**

```
public abstract boolean sybaseContextShouldReportClientMessage(  
    SybaseContext context,  
    NSDictionary clientMessage)
```

Invoked when an exception results from a callback to the `CS_CLIENTMSG_CB` (Sybase ClientMessage callback). Gives the delegate the opportunity to substitute `clientMessage` as the `userInfo` dictionary.

## INTERFACE SybaseContext.Delegate

### sybaseContextShouldReportServerMessage

```
public abstract boolean sybaseContextShouldReportServerMessage(  
    SybaseContext context,  
    NSDictionary serverMessage)
```

Invoked when an exception results from a callback to the CS\_SERVERMSG\_CB (Sybase ServerMessage callback). Gives the delegate the opportunity to substitute *serverMessage* as the userInfo dictionary.

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.