Release Notes:
Enterprise Objects Framework

This file contains release notes for the 1.1 release of the Enterprise Objects Framework. It identifies the differences between this release and the 1.0 release of the Enterprise Objects Framework.

Release 1.1 of the Enterprise Objects Framework fixes a number of problems with the 1.0 release, and substantially improves performance. This release is also 4-way fat; it can be used to develop software for NeXT, Intel, SPARC, and HP-PA machines.

Installation Warning

The Oracle adaptor, formerly used for Oracle 6 databases, is no longer included in this release. The Oracle7 adaptor supports both Oracle v6 and Oracle v7 databases, and should be used for both. When this release of the Enterprise Objects Framework is installed, any applications that use the Oracle adaptor will cease to work until you rebuild the model files used by those applications to use the new Oracle7 adaptor. See problem 50215, below, for more information.

New Features

- The Enterprise Objects Framework is now 4-way fat; you can use it to develop software for SPARC as well as NeXT, Intel, and HP-PA machines.

- The HP-PA version of the Oracle 7 client libraries have been added to the Oracle7 adaptor. In addition, the Oracle7 adaptor has been enhanced so that you can now specify to the Oracle server the character set being used by the client, as well as the language in which you want server error messages to appear. The Oracle7adaptor can now generate SQL*Net v2-style connections strings. For more information on the Oracle7 adaptor, see the Oracle7 Adaptor appendix in the *Enterprise Objects Framework Reference*.

- The version of Foundation that's included with this release includes classes that support multiple threads of execution and the locking of critical regions of code. For more information, see the online class specifications for NSThread, NSLock, NSConditionLock, and NSRecursiveLock as well as the NSLocking protocol specification.

- The version of the Sybase adaptor supplied with this release runs against Sybase System 10. It takes advantage of System 10-specific features including System 10-specific data types and Sybase password encryption, and allows you to specify a client character set and language. It also reads primary key and relationship information from the database when creating new models. For more information on the Sybase adaptor, see the Sybase Adaptor appendix in the *Enterprise Objects Framework Reference*.

- This release now supports NEXTSTEP's Distributed Objects system. In particular, you can now pass subclasses of NSObject from one application to another. For more information, see **UsingDistributedObjects.rtf** in **/NextLibrary/Documentation/NextDev/EnterpriseObjects** and the distributed objects examples in **/NextDev/Examples/EnterpriseObjects**.

- This release is source-code compatible with the Japanese version that runs on the Japanese edition of NEXTSTEP.

New Methods

This section lists methods that have been added since the 1.0 release of the Enterprise Objects Framework.

EOAdaptorChannel

- Two new methods have been added to EOAdaptorChannel that allow default models to have primary keys set, relationships added, and so forth. These methods are:

```
- (NSArray *)describeTableNames
- (EOModel *)describeModelWithTableNames:(NSArray *)tableNames
```

  describeTableNames reads and returns an array of table names from the database. **describeModelWithTableNames:** constructs a default model out of the database's meta data. It also puts the adaptor name and connection dictionary in the new model. **describeModelWithTableNames:** *obsoletes* EOAdaptorChannel's **describeEntities**, **describeAttributesForEntity:**, and **describeRelationshipsForEntity:** methods.

- Two new methods have been introduced to EOAdaptorChannel for use only by its subclasses:

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

+ (NSCalendarDate *)dateForAttribute:(EOAttribute *)attr year:(int)year
month:(unsigned)month day:(unsigned)day hour:(unsigned)hour minute:
```

```
(unsigned)minute second:(unsigned)second zone:(NSZone *)zone;
```

dictionaryWithObjects:forAttributes:zone: is used by AdaptorChannel subclasses to create dictionaries that can be returned from **fetchAttributes:withZone:**. **dateForAttribute:year:month:day:hour:minute:second:zone:** is used by AdaptorChannel subclasses to create a calendar date object to return in an adaptor row. Use of these methods is optional; they were added because they enhance performance.

EOController

- €EOController now has methods that allow you to specify whether or not EOAssociationNotification messages are sent (they are sent by default):

  ```
  - (void)disableAssociationNotification;
  - (void)reenableAssociationNotification;
  ```

  These messages nest; for each **disableAssociationNotification**, you should send a corresponding **reenableAssociationNotification.** The final call to **reenableAssociationNotification** doesn't flush pending notifications to the associations; **[controller redisplay]** must be explicitly called if needed.

- €EOController's new **controller:createObjectFailedForDataSource:** delegate method allows you to respond when the controller is unable to create an object for a data source. If the delegate doesn't implement this method, the controller will display an attention panel to alert the user of the error. Otherwise, the delegate is responsible for notifying the user. This method is declared as follows:

  ```
  - (void) controller:(EOController *)controller
      createObjectFailedForDataSource:dataSource;
  ```

- The following methods were added to EOController to enable key-based in-memory sorting of enterprise objects, without re-fetching them from the data source:

```
- (void)setSortOrdering:(NSArray *)keySortOrderArray;
- (NSArray *)sortOrdering;
- (void)resort;
```

setSortOrdering: allows your code to specify an EOKeySortOrder array to be applied to all records upon a fetch (see **EOKeySortOrdering.h** for more information on EOKeySortOrder). **sortOrdering** returns the array specified with **setSortOrdering:**. **resort** forces the object array to be sorted and redisplayed. Note that the array is automatically sorted after a fetch.

In addition to the three methods listed above, the following controller delegate method has been added:

```
- (void)controller:(EOController *)controller
    sortObjects:(NSMutableArray *)objects;
```

If the delegate implements this method, it's responsible for sorting the given object array. If the delegate doesn't implement this method and a sortOrdering has been specified, the controller sorts the objects itself using **[objects sortUsingKeyOrderArray:[self sortOrdering]]**.

EODatabase

· EODatabase now has a **forgetAllObjects** method that removes all objects from the uniquing table and destroys all of the associated snapshots.

EODatabaseDataSource

·€EODatabaseDataSource now has three new class methods that programmatically name database channels:

```
+ (void)registerChannel:(EODatabaseChannel *)channel
```

```
    forRendezvousWithDatabaseName:(NSString *)aDatabaseName
    contextName:(NSString *)aContextName
    channelName:(NSString *)aChannelName;

+ (void)releaseObjectsWithDatabaseName:(NSString *)aDatabaseName
    contextName:(NSString *)aContextName
    channelName:(NSString *)aChannelName;

+ (EODatabaseChannel *)databaseChannelWithDatabaseName:
    (NSString *)aDatabaseName
    contextName:(NSString *)aContextName
    channelName:(NSString *)aChannelName;
```

registerChannel:forRendezvousWithDatabaseName:contextName:channelName: registers the channel and its parent context and database in the EODatabaseDataSource rendezvous dictionary. If a component is already registered under the specified name, it's replaced with the new component. A counter is associated with each named component: Each time **registerChannel...** or an **init** method is invoked, the counter associated with each of the names used by that method is incremented. When an EODatabaseDataSource is deallocated, it decrements the counters associated with any components that it rendezvoused with at init time.

The **releaseObjectsWithDatabaseName:contextName:channelName:** method decrements the counters associated with each of its name arguments. When a counter reaches 0, its associated name and object are removed from the rendezvous dictionary.

databaseChannelWithDatabaseName:contextName:channelName: returns the database channel registered under the specified database, context, and channel names.

EOQualifier

· There is a new method in EOQualifier that's similar to **qualifierForRow:relationship:**, except it uses the current values of properties in the enterprise object, rather than values in the snapshot:

```
+ (EOQualifier *)qualifierForObject:sourceObject
    relationship:(EORelationship *)relationship;
```

This method creates a qualifier that can be used to fetch the objects that are the destination of an enterprise object's relationship.

EORelationship

· EORelationship has a new method that returns the array of sub-relationships that make up a flattened relationship, or **nil** if the relationship isn't flattened:

```
- (NSArray *)componentRelationships
```

NSNumber

·€NSNumber now has init methods corresponding to the existing autoreleasing factory methods:

```
- initWithChar:(char)value
- initWithUnsignedChar:(unsigned char)value
- initWithShort:(short)value
- initWithUnsignedShort:(unsigned short)value
- initWithInt:(int)value
- initWithUnsignedInt:(unsigned int)value
- initWithLong:(long)value
- initWithUnsignedLong:(unsigned long)value
- initWithLongLong:(long long)value
- initWithUnsignedLongLong:(unsigned long long)value
- initWithFloat:(float)value
- initWithDouble:(double)value
```

```
- initWithBool:(BOOL)value
```

NSObject

- €NSObject now invokes **unableToFaultWithPrimaryKey:entity:databaseChannel:** when a fetch of a to-one fault doesn't return exactly one record. The default implementation of this method raises an error. Clients may override this method to properly initialize the object (or raise).

- To specify behavior when EONull is assigned to a scalar, implement the following method:

  ```
  - (void)unableToSetNullForKey:(NSString *)key
  ```

  Classes can implement this method to define the behavior when EONull is assigned to a property in an enterprise object that requires a C scalar type (such as int or float). One possible implementation is to call **takeValuesFromDictionary:** recursively with a special value object that will return the desired values to assign for the methods **floatValue**, **intValue**, **unsignedIntValue**, and so forth. For an example of how to do this, see **/NextDeveloper/Examples/EnterpriseObjects/Validation.**

NSUtilities

- The declaration of **poseAsClass:** in NSUtilities.h has been changed; it now is declared to return void:

  ```
  + (void)poseAsClass:(Class)class
  ```

NXTableView

· NXTableView now has a method that allows you to set the row height:

```
- setRowHeight:(float) rowHeight
```

Other

· An EOEntity mapping method has been added that allows a class specified by a particular entity to substitute a different class based on its fetched row or primary key values:

```
+ (Class)classForEntity:(EOEntity *)entity values:(NSDictionary *)values;
```

The returned class *must* be either a subclass of the original class, have an entity specified in the model, or implement the **entity** method of the EOEntityMapping protocol. This requirement isn't checked, but will cause insert, deletes, and updates to fail if it's not met.

There are three situations in which this EOClass mapping method is invoked:

1. When an object is created by a data source, usually for insertion. *values* will be nil.
2. When an object referencing this object is fetched and a fault is created. *values* will be the object's primary key.
3. When a row is fetched. *values* will be the fetched row. Note that because of fault creation (#2, above), a class can't rely on always receiving the complete row. To be safe, class selection should be based only on primary key values.

New Behavior

This section identifies behavior that has changed since the 1.0 release of the

Enterprise Objects Framework.

Data Sources

- €If you send **beginTransactionsAutomatically:NO** to an EODatabaseDataSource, an attempt to insert, delete, update or select an object without first beginning a transaction will fail. In the 1.0 release, the **fetchObjects:** method would begin a transaction for you despite the configuration of your data source. This method now issues an error and returns NO. Programs that counted on the 1.0 behavior will no longer work.

- Data sources now always return nil to indicate an error during fetching. In the 1.0 release, either nil or an empty array was returned when an error occurred.

- You can now insert an object into or delete an object from a detail data source even when the object is the destination of a to-one relationship. Insert works only if the corresponding master object property is **nil** or EONull. Delete only works if it isn't. When you delete an object from a detail data source that's the destination of a to-one relationship, the corresponding property in the master enterprise object is set to EONull. This implies that your code now has to watch for situations where the detail object could actually be EONull.

  The client is still totally responsible for assigning the appropriate foreign keys in the master and detail objects.

- **[EODatabaseDataSource coerceValue:forKey:]** no longer returns nil if the key passed in doesn't map to an attribute. If the value corresponds to an attribute in the model, **coerceValue:forKey:** attempts to convert it to the data type specified in the model. It then returns the new representation, or nil if conversion failed (because the input data was invalid). **coerceValue:forKey:** returns the value passed in if the key doesn't correspond to an attribute.

Controllers

- The delegate of an EOController now receives **controllerWillFetch:** before **controllerWillDiscardEdits:** and **controllerWillDiscardOperations:**.

- Because a controller delegate couldn't adequately specify what to do in response to a **will...Object:inDataSource:** message, **controller:willInsertObject:inDataSource:**, **controller:willDeleteObject:inDataSource:**, and **controller:willUpdateObject:inDataSource:** now return **EODataSourceOperationDelegateResponse**. This enumerated type is defined as follows:

```
typedef enum {
    EODiscardDataSourceOperation = 0, /* NO response in EOF 1.0 */
    EOPerformDataSourceOperation,     /* YES response in EOF 1.0 */
    EOContinueDataSourceOperation,
    EORollbackDataSourceOperation,
} EODataSourceOperationDelegateResponse
```

EOPerformDataSourceOperation means that the operation should proceed.
EODiscardDataSourceOperation means that the operation shouldn't be performed, and should be discarded from the operation stack.
EOContinueDataSourceOperation means that the operation shouldn't be performed but should be left on the operation stack to be performed on the next save.
EORollbackDataSourceOperation menas that the current **saveToDataSource** operation should be aborted and all changes rolled back.

Adaptor Channels

- The specification for EOAdaptorChannel's

adaptorChannel:willFetchAttributes:withZone: method has changed. This method doesn't allow changes to the attribute array. The new specification for this method is:

```
- (NSMutableDictionary *)adaptorChannel:channel
    willFetchAttributes:(NSArray *)attributes
    withZone:(NSZone *)zone
```

Database Channels

- The specification for EODatabaseChannel's **databaseChannel:willSelectObjectsDescribedByQualifier:fetchOrder:** method has changed. This method doesn't allow changes either to the qualifier or to the fetch order. The new specification for this method is:

```
- (BOOL)databaseChannel:channel
    willSelectObjectsDescribedByQualifier:(EOQualifier *)qualifier
    fetchOrder:(NSArray *)fetchOrder
```

Oracle7 Adaptor

- The Oracle7 adaptor now supports both SQL*Net v1 and v2-style connection strings. The version of SQL*Net used is determined by the connection string you provide (whether through the login panel or through the connection dictionary in your model file).

  To use SQL*Net v1, supply a user name, password, host machine name, and server ID. This results in a v1-style connection string of the form ª*userName*/*password*@T:*hostMachine*:*serverId*º.

  To use SQL*Net v2, supply a serverId (a SQL*Net V2 style server string), user name, and password, but omit the host machine name. This results in a v2-style connection string that has the form ª*userName*/*password@serverId*º. If you want to

use a custom connection string, omit values for all of the keys except **serverId** in your connection dictionary. You can then use the Server ID field in the Oracle login panel to supply your own connection string.

Other

- NXImageView flags are now archived consistently between Intel, NeXT, and HP-PA machines. As a result, if your applications have archived NXImageViews, their settings may not be correctly restored after this release of the Enterprise Objects Framework is installed. Specifically, Intel users installing this release will lose the border and editability flags on their NXImageViews. To fix them, reset these attributes in Interface Builder and save the nib file.

- An external query specified for an entity must return columns in alphabetical order for the entity's attributes. In EOF 1.0, the order used was less predictable (attributes used for locking sorted alphabetically, followed by the components of the primary key, followed by the class properties sorted alphabetically).

- There is now a dwrite that you can use to specify the location of the Sybase adaptor's interfaces file. If you have installed your Sybase server somewhere other than **/usr/sybase**, specify the location of the interfaces file with:

```
dwrite SybaseAdaptor SybaseInterfacesFile path_to_interfaces_file
```

Other Notes

NXZones and NSZones

To avoid compatibility problems, in this release of the Enterprise Objects Framework the NSZone structure is identical to the structure used for NXZones. NSZones and NXZones can be used interchangeably by casting as needed.

Problems Fixed in This Release

This section identifies those problems that have been fixed in this release of the Enterprise Objects Framework.

Enterprise Objects Framework

Reference: 41356

Problem: There should be a way to turn off autodisplay.

Description: Given a summary table view and a couple of Forms that display details for the selected row, selecting a new row in the table view causes a really slow update of the Form's cells. The window is currently flushing after each FormCell changes its value. It might be faster to disable FlushWindow, update the Forms (and other UI elements), and then flush the window after all the drawing is done.

Fix: EOController now has two additional methods (**disableAssociationNotification** and **reenableAssociationNotification**) to disable or enable the sending of EOAssociationNotification messages.

Reference: 44839

Problem: Interface Builder sorts attributes differently than EOModeler.

Description: If you have an existing entity defined in the EOModeller and you add a new attribute, after updating the display, the new attribute appears in alphabetical order in EOModeller. However, the new attribute appears at the *bottom* of the list in the EOController inspector in Interface builder.

Fix: Interface Builder and EOModeler now display attributes sorted alphabetically.

Reference: 45878

Problem: EODatabase needs a **forgetAllObjects** method.

Fix: EODatabase now has a **forgetAllObjects** method that removes all objects from the uniquing table and destroys all of the associated snapshots.

Reference: 46623

Problem: Forced transactions on select.

Description: When you send **beginsTransactionsAutomatically:** to the data source with NO as its argument, you can control the begin and commit/rollback for insertions, updates, and deletions. However, the Framework still issues BEGIN TRANSACTION and COMMIT TRANSACTION statements around each SELECT statement. This interferes with stored procedures which embed the SELECT INTO phrase in the SQL, since this can't be executed inside a transaction.

Fix: The Framework no longer begins a transaction for you when **fetchObjects:** is invoked if **beginsTransactionsAutomatically** is set to NO.

Reference: 47227

Problem: Failed one-to-one faults shouldn't raise.

Description: In EOF 1.0, if the relationship is optional (and non-existent), an exception is raised that can't be caught because the exception can be called at any time based on user actions. There is no way to ignore the failed relationship and continue with the fetch.

Fix: If the foreign key for a to-one relationship in the sorce object is EONull, the relationship property is automatically set to **nil**. If the foreign key is non-null and the to-one fault fails, NSObject now invokes **unableToFaultWithPrimaryKey:entity:databaseChannel:** on the faulted object.

Reference: 47358

Problem: There is no way to programmatically name EODatabaseChannels.

Description: In InterfaceBuilder, there is an inspector for EOControllers that allows the programmer to type in a database channel name, a context name, and a database name, but there is no way to set these names programmatically.

Fix: EODatabaseDataSource now has three new class methods that programmatically name database channels: **registerChannel:forRendezvousWithDatabaseName:contextName:channelName:**, **releaseObjectsWithDatabaseName:contextName:channelName:**, and **databaseChannelWithDatabaseName:contextName:channelName:**.

Reference: 47719

Problem: A controller should ask its delegate about a fetch before discarding.

Description: In the 1.0 release, a controller throws away all pending changes and the contents of its undo stack before invoking **controllerWillFetch:**. If the

delegate returns NO, the controller doesn't fetch; but it has lost the pending changes and the contents of its undo stack.

Fix: The delegate of an EOController now receives **controllerWillFetch:** *before* **controllerWillDiscardEdits:** and **controllerWillDiscardOperations:**.

Reference: 47734

Problem: EODatabaseChannel's **databaseChannel:willSelectObjectsDescribedByQualifier:fetchOrder:** method shouldn't allow the qualifier to change.

Description: **databaseChannel:willSelectObjectsDescribedByQualifier:fetchOrder:** is incorrectly documented in the header file as allowing the delegate to change the qualifier and fetch order. This method doesn't allow either the qualifier or the fetch order to change.

Fix: This method no longer allows changes to either the qualifier or the fetch order.

Reference: 47779

Problem: **fetch** may return YES even when an error occured during the fetch.

Description: EOController's **fetch** method doesn't check the return value of EODatabaseDataSource's **fetchObjects** method for **nil**. Consequently, it returns YES even when a data source error occured during the fetch.

Fix: Data sources now always return nil to indicate an error during fetching, and **fetch** then returns NO.

Reference: 47784

Problem: The NXTableValue protocol incorrectly declares **isEqual:** and **setValueFrom:**

methods.

Description: The NXTableValue protocol (declared in **NXTableProtocols.h**) declares **isEqual:** and **setValueFrom:** methods. It shouldn't; these methods aren't supported.

Fix: These methods have now been removed from the NXTableValue protocol.

Reference: 47855

Problem: Associations to Cells in deferred windows sometimes don't work.

Description: Deferred windows containing Cells associated with an EOController must be displayed during or immediately after being loaded from their nib file.

Fix: This has been fixed.

Reference: 47962

Problem: NSArray's **insertObjectAtIndex:** can put up a panel that a delegate can't override.

Description: All potential panel presentations are supposed to be catchable by the controller delegate, but there isn't a **controller:dataSourceFailedToCreateObject:** delegate message, so the controller always puts up a panel.

Fix: EOController now has a delegate method named **controller:createObjectFailedForDataSource:** that can be implemented to respond when the controller is unable to create an object for a data source.

Reference: 48431

Problem: Should report ªObject doesn't respond toº instead of ªUnknown error 1000002º.

Description: When an unsupported method is sent to a Framework object, rather than the typical ª<object> doesn't respond to <method>º message, something along the lines of ªUnknown error code 100002 in NXReportErrorº is generated.

Fix: The Framework now generates an error message that contains both the name of the exception and the reason for the exception, along with the error code.

Reference: 48612

Problem: New adaptor API is needed to read database meta data.

Description: This new adaptor API would allow default models to have primary keys set, relationships added, and so forth. There is also a potential performance win of reading a model all at once.

Fix: EOAdaptor has two new methods: **describeTableNames** and **describeModelWithTableNames:** that allow adaptors to read database meta data.

Reference: 48803

Problem: **will...InDataSource:** delegate messages have inadequate return values.

Description: There is a problem when implementing validation in **controller:willInsertObject:inDataSource:**Đreturning NO doesn't put the object on the failure stack, nor does it rollback the transaction. It throws the buffered insert operation away, and continues.

Fix: The **will...InDataSource:** now return an **EODataSourceOperationDelegateResponse**, instead of a **BOOL.**

Reference: 48811

Problem: NXTableView stays scrolled after linking an association.

Description: When you link an association to a table view, if the table view has to scroll to make the connection it stays scrolled both in the nib file and when the application runs.

Fix: Scrolling in NXTableViews now works correctly.

Reference: 48947

Problem: An interface is needed to allow the setting of the row height for a table view.

Fix: NXTableView now supports **setRowHeight:**.

Reference: 48948

Problem: EODatabaseDataSource's **coerceValue:forKey:** incorrectly returns **nil**.

Description: **[EODatabaseDataSource coerceValue:forKey:]** will return **nil** if the key passed in doesn't map to an attribute. Instead, it should pass the value through. Clients of a data source want to perform any available coercion on a set of values and pass the rest through to later stages of validation. If **nil** is returned for values the data source doesn't know about, the caller can't easily tell if this indicates an error (an attribute that failed conversion) or just the use of a key not in the entity.

Fix: **coerceValue:forKey:** no longer returns **nil** if the key passed in doesn't map to an attribute. Instead, it returns the uncoerced (unprocessed) value.

Reference: 48961

Problem: There should be a method on EORelationship that returns the flattened relationship array.

Description: Flattened relationships consist of an array of two or more sub-relationships. It would be nice if this array could be made available to clients. It's very useful when attempting to generically handle foreign key assignment, and such.

Fix: EORelationship now supports the **componentRelationships** method.

Reference: 49046

Problem: EOControllers should support sorting.

Description: EOControllers should support key-based in-memory sorting of enterprise objects, without re-fetching them from the data source.

Fix: EOControllers now support key-based sorting of enterprise objects with the **setSortOrdering:**, **sortOrdering**, and **resort** methods.

Reference: 49479

Problem: NXImageView flags don't carry between Intel and NeXT/HP-PA.

Description: If I set an ImageView to have no border on an Intel machine, it will still have a border when I load it up on an NeXT or HP-PA machine. The flags struct is being archived incorrectly.

Fix: This has been fixed so that the flags are archived consistently. If your applications have archived NXImageViews, their settings may not be correctly restored after this release of the Enterprise Objects Framework is installed. However, if you re-set the flags in the appropriate nib files and re-archive the NXImageView, it will then operate correctly.

Reference: 50134

Problem: Undo doesn't work for controllers not created in Interface Builder.

Fix: This has been fixed.

Reference: 50402

Problem: EOControlAssociations don't work with non-text controls.

Description: Upon receiving an action message from its target, an EOControlAssociation asks for **nextText**. This is a problem for non-text controls (like sliders).

Fix: This has been fixed.

Reference: 50632

Problem: Resorting on key with EONull values raises an exception.

Description: In the pre-release of the Enterprise Objects Framework release 1.1, controller in-memory sorting could raise if the columns being sorted contained EONulls.

Fix: This has been fixed.

Reference: 50739

Problem: EOAdaptorChannel's **adaptorChannel:willFetchAttributes:withZone:** delegate method was incorrectly specified to allow the attribute array to be changed.

Fix: The API has been changed so that the attribute array is declared as an NSArray instead of an NSMutableArray.

Reference: 50939

Problem: Calling EOEntity's **removeAttributeNamed:** sometimes caused a crash with the message: "objc: FREED(id): message setEntity: sent to freed object=0x1a06c8"

Fix: This has been fixed.

Reference: 50945

Problem: Error when selecting multiple objects at end and refetching fewer objects.

Description: Fetching objects, making a selection near the end, and then fetching a smaller set of objects (perhaps by setting a qualifier) may result in an error being raised of the form:

```
*** -objectAtIndex:: index (3) beyond bounds (0)
```

Fix: This has been fixed.

Reference: 51211

Problem: Constraints on model object names don't allow kanji characters.

Fix: The naming constraints on model objects (Entity, Relationship, and Attribute) have been relaxed to allow non-ascii characters. These names can contain any character except one of the following:

```
!"$%&'()*+,-./:;<=>?[\]^`{|}~
```

Names may also begin with a $ but may not contain a $ inside the name.

Oracle Adaptors

Reference: 46780

Problem: The Oracle7 adaptor doesn't support custom values.

Fix: Custom value support has been added to the Oracle7 adaptor.

Reference: 49800

Problem: The Framework needs to provide SQL*Net v2 capability.

Fix: The Oracle7 adaptor now supports both SQL*Net v1 and v2-style connection strings. The version of SQL*Net used is determined by the connection string you provide (whether through the login panel or through the connection dictionary in your model file).

To use SQL*Net v1, supply a user name, password, host machine name, and server ID. This results in a v1-style connection string of the form ª*userName*/*password*@T:*hostMachine*:*serverId*º.

To use SQL*Net v2, supply a serverId (a SQL*Net V2 style server string), user name, and password, but omit the host machine name. This results in a v2-style connection string that has the form ª*userName*/*password*@*serverId*º. If you want to use a custom connection string, omit values for all of the keys except **serverId** in your connection dictionary. You can then use the Server ID field in the Oracle login panel to supply your own connection string.

Reference: 51023

Problem: Trying to retrieve a LONG from an Oracle database can cause an infinite loop.

Description: In trying to retrieve a LONG (but not a LONG RAW) from an Oracle database, the app can be put into an infinite loop. The problem occurs both when setting the value to an NSString or to a custom value.

Fix: This has been fixed in the Oracle7 adaptor.

Reference: 51626

Problem: There is no reasonable way for Oracle adaptor users to specify a non-default character set.

Fix: There is now a new connection dictionary key for the Oracle7 adaptor: NLS_LANG. This key allows you to set the Oracle NLS_LANG environment variable. NLS_LANG declares to the Oracle server the character set being used by the client, as well as the language in which you want server error messages to appear. The format is as follows:

language_territory. *characterSet*

For example, supplying the value **japanese_japan.jeuc** for the NLS_LANG key tells the server that the language is Japanese, the territory is Japan, and the character set is **jeuc** (**japanese_japan.jeuc** is the default for Japanese systems). See your Oracle documentation for a complete list of types available for this field.

To add the NLS_LANG key and a value to your connection dictionary, you must manually edit your model file.

Sybase Adaptor

Reference: 43430

Problem: The Framework should create default relationships for new models with Sybase databases.

Description: The common/foreign key information in Sybase should be used to create default relationships for a new model.

Fix: The Sybase adaptor now uses the new EOAdaptor methods that read a database's meta data to construct a model that has default relationships filled in.

Reference: 46674

Problem: There should be a dwrite to specify an alternate location for **/usr/sybase/interfaces**.

Description: This dwrite would be useful for people that install a sybase server somewhere besides **/usr/sybase** (such as their home directory).

Fix: Use the following to specify an alternate location for the sybase server's interfaces file:

```
dwrite SybaseAdaptor SybaseInterfacesFile path_to_interfaces_file
```

EOModeler

Reference: 49076

Problem: EOModeler overwrites links with plain files when saving models.

Description: If you have a link to a model in **~/Library/Models**, when you select the link (or double-click the corresponding EOController in Interface Builder), edit the model, and save it, EOModeler replaces the link with a copy of the file. You now have two models: the edited one in **~/Library/Models**, and the unedited model in its original location.

Fix: EOModeler now handles links correctly.

Reference: 50207

Problem: EOModeler's Model Browser tries to access objects that have been freed.

Description: If you drag an item to the Model Browser and then delete it in the Model Editor, EOModeler will crash if you go back to the Model Browser and either try to change the sort order or drag the first column to the left.

Fix:

EOModeler's Model Browser no longer tries to access freed objects.

Reference: 50244

Problem: A remote connection from an Intel computer to EOModeler running on a RISC computer can cause the RISC computer's window server to hang.

Description: This problem happens between any Intel computer running NEXTSTEP 3.3 and NEXTSTEP Developer 3.3 and a SPARC or HP-PA computer. From the Intel computer, run EOModeler remotely on the RISC computer. In an Oracle7 database, browse an entity with an attribute that's null. If you then drag the tear-off into the Model Browser well, both EOModeler and the RISC computer's window server hang.

Fix: Remote connections to RISC computers now work correctly.

Reference: 51201

Problem: Inspector becomes confused after changing class properties.

Description: Sometimes setting certain properties to be class properties in the Entity inspector would cause other properties to later show up in the inspector with a diamond, even though they had not been marked that way and were not listed in the model file as class properties.

Fix: This has been fixed.

Foundation

Reference: 48449

Problem: Incompatibilities between between the declarations of **poseAsClass:** in NSObject and NSUtilities.

Description: **NSObject.h** declares **poseAsClass:** as `+(void)poseAsClass:(Class)aClass', while NSUtilities.h declares it as `+poseAsClass:(Class)class'.

Fix: The declaration of **poseAsClass:** in NSUtilities.h has been changed; it now is declared to return void.

Reference: 49826

Problem: **stringWithCString:length:** does the wrong thing when default encoding is Kanji.

Description: **[NSString stringWithCString:length:]** instantiates from NSInlineCString (which is for 8-bit encodings) even if the default encoding is 16-bit, while **[NSString initWithCString:length:]** returns the correct NSUnicodeString instance.

Fix: **[NSString stringWithCString:length:]** now returns the correct NSUnicodeString instance.

Reference: 49827

Problem: smallestEncoding doesn't return the smallest encoding.

Description: NSString's **smallestEncoding** is supposed to return the most space efficient encoding for the given string, but it seems to currently always return the same as fastestEncoding, which is the ªnativeº encoding of the string.

Fix: **smallestEncoding** now returns the correct encoding.

Reference: 49846

Problem: NSStrings, NSDatas, and NSNumbers are leaked when sent over old DO.

Description: Temporary encoding objects are created for instances of NSString, NSData, and NSNumber when they are passed via DO. These temporary objects copy the real objects but never release them.

Fix: Each of the temporary encoding classes now release the copied instances.

Reference: 51728

Problem: Identical NSDates compare as NSOrderedAscending instead of NSOrderedSame.

Fix: This has been fixed.

Known Problems in Release 1.1

Enterprise Objects Framework

These problems exist in this release of the Enterprise Objects Framework:

Reference: 41338

Problem: Comparing equivalant qualifiers using **isEqual:** indicates that they are not equivalent.

Description: When two qualifiers are compared using **isEqual:**, their identifiers are compared instead of their contents. Thus, **isEqual:** returns NO even if the two

qualifiers have the same contents.

Workaround: Send **expressionValueForContext:** to each qualifier to obtain each qualifier's contents as an NSString, and then compare the two NSStrings using **isEqual:**. For example:

```
[[qualifier1 expressionValueForContext: nil] isEqual:
     [qualifier1 expressionValueForContext: nil]]
```

Reference: 46679

Problem: Instance variables are not documented.

Description: Instance variables of Enterprise Objects Framework classes should not be accessed. These will be made private in a later release.

Workaround: None.

Reference: 46967

Problem: **deleteObject:** fails, but claims to succeed.

Description: The use of constraint enforcement techniques in the database instead of the model may cause the Enterprise Objects Framework to become out of sync. This becomes a bigger problem if the Framework is executing several operations as part of a transaction and the transaction is rolled back by a trigger rule.

Workaround: None.

Reference: 47196

Problem: The Enterprise Objects Framework does not perform automatic character set conversions.

Description: The Framework sends data to the database in the default character encoding. This can be a problem when the following is true:

· non-ASCII characters are being stored in the database

· the data is accessed by machines using a different character encoding than the
default NEXTSTEP encoding

· the database server does not provide character conversion from the default NEXTSTEP encoding to the character encoding used by the database

This situation may commonly exist in Europe, where accented characters are used which need to be stored in the iso_8859_1 encoding.

Workaround: In order to implement the necessary conversion from NEXTSTEP to iso_8859_1 encoding, implement a subclass of NSString which will perform the necessary conversion to and from the NEXTSTEP encoding, and specify that subclass as a custom class on any attributes which contain character data that may need to be converted.

Reference: 47217

Problem: EOController doesn't retain its delegate.

Description: EOController doesn't retain its delegate, but everything else in the Framework does (this is likely to change in a later release of the Framework).

Reference: 47563

Problem: There needs to be a way to notify a controller than an enterprise object has been edited.

Description: There should be a way to notify a controller that you have edited an enterprise

object behind the controller's back. EOController provides a method that does something along these lines (**setValues:forObject:**), but it can be inconvenient and unnatural to use.

Workaround: If you change an enterprise object outside of the controller (for instance, by sending it a message such as **setSalary:** directly) you can cause the controller to buffer it for update by sending it **setValues:forObject:** and supplying an empty dictionary:

```
[myController setValues:[NSDictionary dictionary]
    forObject:object];
```

Reference: 47754

Problem: EOController's **keys** method isn't very useful.

Description: **[EOController keys]** returns a combined list of the keys from each association, including duplicates. It doesn't return other keys supported by the data source, nor does it return keys that have been entered into Interface Builder's attribute inspector.

It should at least remove the duplicates.

Workaround: None.

Reference: 47832

Problem: Unable to update custom values (such as NXImages) in the database.

Description: Custom value classes must implement **isEqual:** if they are to be used for locking.

Workaround: Don't mark as ªused for lockingº those attributes in the model with a custom data type unless the custom value class implements a value-based **isEqual:** method.

Reference: 48261

Problem: Triggering a fault can cause your application to hang.

Description: **objectFaultWithPrimaryKey:** does not check if the object is already in the uniquing tables. Triggering a fault that represents an object that's already in memory may cause an infinite loop.

Workaround: Check the uniquing tables for the existence of an object before constructing a fault for it.

Reference: 48801

Problem: There needs to be a way to empty an EOController.

Workaround: Set the data source to **nil** and then back.

Reference: 48949

Problem: **convertValue:forKey:** returns an NSNumber with the value 0 when number conversion fails.

Description: If you use **coerceValue:forKey:** for an NSNumber attribute and supply something that can't be coerced to a number ("abc", for instance), you get back an NSNumber with the value 0, rather than nil (to indicate the error).

Workaround: Perform your own validation of number strings before calling **coerceValue:forKey:**.

Reference: 49659

Problem: It can be difficult to use dates in qualifiers.

Description: The following qualifier:

```
startDate = [NSCalendarDate dateWithYear:1994 month:11
    day:27 hour:0 minute:0 second:0
    timeZone:[NSTimeZone timeZoneWithAbbreviation:@"PST"]];

qualifier = [[EOQualifier alloc] initWithEntity:[ds entity]
     qualifierFormat:@"%A < %@", @"DATE", startDate];
```

Generates this (bad) SQL (the date is dropped from the SQL):

```
"... where DATE <"
```

It turns out that **expressionValueForContext:** isn't (and can't be) implemented for NSDate.

Workaround: Use **formatValue:forAttribute:**, as shown here:

```
EOAdaptor *adaptor = [[[[ds databaseChannel] adaptorChannel]
    adaptorContext] adaptor];
qualifier = [[EOQualifier alloc] initWithEntity:[ds entity]
    qualifierFormat:@"%A < %@", @"NAME",
    [adaptor formatValue:startDate forAttribute:dateAttr]];
```

Contrary to the documentation, you *can* invoke **formatValue:forAttribute:** directly.

Reference: 50097

Problem: Master-peer with a flattened relationship can generate an invalid ORDERBY clause.

Description: Setting the fetch order on an EODatabaseDatasource that is the peer in a master-peer relationship will generate invalid SQL if the relationship has been flattened.

Workaround: None.

Reference: 50648

Problem: EORelationship's **addJoin:** method fails if the relationship isn't part of an entity.

Description: You cannot add a join to a relationship unless the relationship has either received a **setEntity:** message with a valid entity, or been added to an entity.

Reference: 50771

Problem: Selecting multiple rows in an NXTableView always scrolls to the first selected row.

Description: Adding to the selection in a TableView causes it to scroll back to the first selected row.

Workaround: Write a subclass of EOColumnAssociation that only scrolls when none of the selected rows are visible.

Reference: 51266

Problem: Changes to relationship objects in **willSaveEdits:** aren't undone.

Description: Additional changes returned from **controller:willSaveEdits:toObject:** aren't undone by **undo**.

Workaround: None.

Reference: 51338

Problem: **runLoginPanel** doesn't return **nil** if you cancel the login panel.

Description: **runLoginPanel** doesn't return **nil** when the cancel button is pressed on the adaptor's login panel. Instead, a dictionary is returned which contains the values from the adaptor's connection dictionary.

Reference: 51347

Problem: The size of the EOController object changed in EOF 1.1 prerelease.

Description: The size of the EOController object was inadvertantly changed for the pre-release of the Enterprise Objects Framework release 1.1, thereby breaking binary compatability with the 1.0 release for applications that create subclasses of EOController. This has been fixed in this release of the Enterprise Objects Framework, thereby making it binary compatible with the 1.0 release but incompatible with the 1.1 pre-release.

Reference: 51708

Problem: Distinct selections don't always work.

Description: Qualifiers with **usesDistinct** set to YES will not be distinct when used with data sources that have auxiliary qualifiers set.

Workaround: Have a delegate on the EODatabaseChannel wait for the **databaseChannel:willSelectObjectsDescribedByQualifier:fetchOrder:** method. Have the delegate invoke **setUsesDistinct:**YES on the qualifier that is passed into this method.

Reference: 51743

Problem: Objects that use dates in the primary key aren't always found in the uniquing table.

Description: **objectForPrimaryKey:** may fail to find objects that have a date as the primary key.

Workaround: Make sure that the date used in the lookup has the same format string set as the one set by the corresponding EOAttribute in the model.

Reference: 51768

Problem: Unarchiving EONulls produces distinct instances.

Description: EONull may result in multiple instances after unarchiving. If the EONull object is archived to a file, then unarchiving the file will create a new instance of EONull, one that is not returned by **[EONull null]**.

Workaround: Implement the following category on EONull:

```
@implementation EONull (Unarchiving)
- (id) awakeAfterUsingCoder:(NSCoder *)coder
{
    EONull *oneTrueNull = [EONull null];

    if (self == oneTrueNull)
        return self;

    [self autorelease];
    return oneTrueNull;
}
@end
```

Sybase Adaptor

These problems exist in the Sybase adaptor supplied with this release of the Enterprise Objects Framework:

Reference: 47202

Problem: Conversion from money to double doesn't always work.

Description: The sybase **dbconvert** function doesn't correctly convert from money to double (it introduces extra precision at low end). For instance, the value `2078927178.0000' is converted as `2078927178.0000002384185791'.

Workaround: Use a custom value class for money.

Reference: 48235

Problem: Sybase adaptor ignores millisecond portion of datetime values

Description: The Sybase adaptor represents dates in a Sybase datetime column as NSCalendarDate objects which don't keep millisecond values. Consequently, the millisecond portion of a date stored in a Sybase datetime column isn't fully represented in EOF when using the Sybase adaptor. This causes some updates to fail since a qualifier containing a date value that was read in from the database may not be precise enough to match against the database column if its millisecond value was non-zero.

Workaround: None.

Reference: 48574

Problem: There is no way to suppress the alert panel during a failed login attempt

Description: The **willReportError:** delegate method on the Sybase Adaptor is not called when there are errors while attempting to login. These error messages can only be suppressed by invoking the following class methods on the SybaseAdaptor:

```
[SybaseAdaptor logsMessages:NO]
    while ( ![adaptor hasValidConnectionDictionary] &&
       ![adaptor runLoginPanelAndValidateConnectionDictionary] );
[SybaseAdaptor logsMessages:YES].
```

Workaround: See the description, above.

Oracle7 Adaptor

These problems exist in the Oracle7 adaptor supplied with this release of the Enterprise Objects Framework:

Reference: 47202

Problem: Conversion from money to double doesn't always work.

Description: The sybase **dbconvert** function doesn't correctly convert from money to double (it introduces extra precision at low end). For instance, the value `2078927178.0000' is converted as `2078927178.0000002384185791'.

Workaround: Use a custom value class for money.

Reference: 50553

Problem: The Oracle7 adaptor should ready primary key constraint information.

Description: The Oracle7 adaptor does not take advantage of information available in the database to set the primary key or create relationships when creating new models.

EOModeler

Reference: 49811

Problem: Interface Builder sometimes doesn't notice changes made in EOModeler.

Description: The following procedure illustrates the problem:

1. Create a project.
2. Open the project's nib file and change file's owner.
3. Create a model and save it in **~/Library/Models**.
4. Drag an entity into the nib file.
5. Add the model to the project.
6. Double-click the controller in the nib file to open the model.
7. Add a relationship to the model, then save it.
8. Go back to the nib file; the relationship doesn't show up, and Interface Builder hasn't marked the document as having changed.

Workaround: Quit and restart Interface Builder, then re-open the offending nib file.

Reference: 51650

Problem: Tear-offs don't keep up with the arrow keys in the model editor.

Description: If you open up a **.eomodel** file, select an entity, and press the down or up arrow keys a few times, you will notice that the property names in the tear-offs are one step behind the actual selection (the inspector does display correctly, however).

Foundation

Reference: 46032

Problem: The NSAssert macros have counterintuitive behavior with respect to the

NS_BLOCK_ASSERTIONS flag.

Description: Previously, -DNX_BLOCKASSERTS turned assertions off when compiling. Now, assertions are turned on when -DNS_BLOCK_ASSERTIONS is supplied.

Reference: 51800

Problem: NXAutoreleaseConnection's **connectToName:onHost:** returns a retained object.

Description: If the object on the server side is a subclass of NSObject, this method should return an autoreleased proxy.

Workaround: Clients should release any NSObject proxies that are returned by this method.

Other

Reference: 43200

Problem: Debugging tools are needed.

Description: There are no profiling libs for the 1.1 release of the Enterprise Objects Framework. However, NeXT plans to provide a profiling tool via NeXTanswers.

Locating the Documentation for this Release

This release of the Enterprise Objects Framework consists not only of the classes, protocols, types, and constants that make up the Framework, but also a version of the Foundation Kit (a set of basic utility classes that enhance consistency and portability) and a version of

Interface Builder that has been enhanced to allow you to build Enterprise Objects Framework applications (this is the same version that's shipped with NEXTSTEP Developer version 3.3). Various forms of documentation for each of these products is supplied with this release. The purpose of this section is to help you locate the appropriate documentation for each product supplied with this release of the Enterprise Objects Framework.

Foundation Kit Documentation

All Foundation Kit documentation can be found online, in the following directory: **/NextLibrary/Documentation/NextDev/Foundation**. **IntroFoundation.rtfd** contains an **i**ntroduction to the Foundation Kit. Additional concepts are in the documentation provided for each Foundation Kit class. In particular, look over the documentation for **NSException**; this discusses how exceptions are raised and handled when working with the Foundation Kit.

Reference documentation for all Foundation Kit components can be accessed through Header Viewer. Use Header Viewer©s Preferences menu to add **/NextLibrary/Documentation/NextDev/Foundation** to Header Viewer's list of documentation directories.

Enterprise Objects Framework Documentation

All Enterprise Objects Framework documentation can be found online, in the following directory: **/NextLibrary/Documentation/NextDev/EnterpriseObjects. Guide** contains the *Enterprise Objects Framework Developer's Guide* (this manual is also shipped in printed form with the developer package). While the information in the *Developer's Guide* is accurate, new functionality described in this release note has not been incorporated into either the on-line or the printed version of the *Developer's Guide*. Thus, this release note supplements the *Enterprise Objects Framework Developer's Guide*. The *Enterprise Objects Framework Reference* (which is available on-line only; see the following paragraphs), has been updated with the material in this release note.

The *Enterprise Objects Framework Reference* is divided into three major groupings: the documentation for those classes, protocols, types, and constants of the Access Layer (in **Reference/1_AccessLayer**), the documentation for those classes, protocols, types, and constants that make up the Interface Layer (in **Reference/2_InterfaceLayer**), and the documentation for those classes, protocols, types, and constants that support the additional user interface objects supplied with the Framework (in **Reference/3_UIObjects**). **Reference** also contains directories for the two reference appendices: **_ApA_Oracle7Adaptor** and **_ApB_SybaseAdaptor**.

Reference documentation for all Enterprise Objects Framework components can be accessed through Header Viewer. Use Header Viewer's Preferences menu to *individually* add the following three directories from **/NextLibrary/Documentation/NextDev/EnterpriseObjects** to Header Viewer's list of documentation directories (if you select all of them and try to add them, HeaderViewer won't correctly locate any of the Enterprise Objects Framework documentation):

- **Reference/1_AccessLayer**
- **Reference/2_InterfaceLayer**
- **Reference/3_UIObjects**

Similarly, *individually* add the following two files from **/NextDeveloper/Headers** to Header Viewer's list of header files (if you select both of them and try to add them, HeaderViewer won't correctly locate either file):

- **eoaccess/eoaccess.p**
- **eointerface/eointerface.p**

Documentation Feedback

Your comments on our documentation are especially valuable. Please send electronic mail with your comments and suggestions to **techpubs_feedback@next.com**. (If you include

EODoc: as the first part of your message's subject, we'll be able to identify it more quickly.)