EODistribution Framework

Java API Reference

The EODistribution Layer

Package:	com.apple.client.eodistribution (client side) com.apple.yellow.eodistribution (server side)

Introduction

The EODistribution layer is used in Java Client applications. It consists of two parts: a Yellow Box framework for the server and a Java package for the client. The EODistribution (or, simply, “distribution”) layer performs by-copy object distribution and synchronization. It is responsible for synchronizing the states of the object graphs on the client and on the application server. The distribution layer handles communication over a “channel” (which use transports such as HTTP or CORBA) and moves properties in both directions, that is, as objects are fetched and changes are saved. It encodes and decodes objects as they travel back and forth over the channel.

The classes in the server side of the EODistribution layer are provided by the EOJavaClient framework, and server side APIs are available in both Objective-C and Java (the Java package for the server side APIs is com.apple.yellow.eodistribution). The classes on the client side are implemented in pure Java and live in the com.apple.client.eodistribution package.

The following table summarizes each class in the EODistribution layer:

Class	Client	Server	Description
EODistributedDataSource	X		Fetches data using an EOEditingContext on the client as its source of objects.
EODistributedObjectStore	X		Handles interaction with the distribution layer’s channel, incorporating knowledge of that channel so it can forward messages it receives from the server to its editing contexts and forward messages from its editing contexts to the server.
EODistributionChannel	X		Abstract class for distribution channels.
EODistributionContext		X	Encodes data to send to the client and decodes data it receives from the client; also tracks and communicates any changes on the server object graph to the client.
EOHTTPChannel	X		Implements a distribution channel using HTTP as the transport.
WOJavaClientApplet		X	Used to download and create the applet on the client.

In addition, the utility class EOAccessAdditions declares a number of methods that return client-specific information stored in model files.

EOAccessAdditions

Inherits from:	NSObject
Package:	com.apple.yellow.eodistribution

Class Description

This class consists of a number of static methods to augment EOEntity and EOEntityClassDescription (both declared in EOAccess). These methods return various properties bound to the client-side class that corresponds to the supplied EOEntity or EOEntityClassDescription object. The information returned by these methods is stored in your model file. To change it, use the EOJavaClientExtensions bundle for EOModeler.

You never create or use an instance of EOAccessAdditions. Rather, send messages directly to the class.

Static Methods

clientAttributeKeysForClassDescription

```
public static NSArray clientAttributeKeysForClassDescription(
   com.apple.yellow.eocontrol.EOClassDescription aClassDescription)
```

Returns an array containing the names of those attributes that are bound to the client-side class that corresponds to the *aClassDescription*'s EOEntity.

See Also: `clientClassPropertiesForEntity`

clientClassNameForEntity

```
public static String clientClassNameForEntity(
   com.apple.yellow.eoaccess.EOEntity anEntity)
```

Returns the name of the client-side enterprise object class associated with *anEntity*. If no client-side class name has yet been registered for *anEntity*, this method returns the name of the receiving class (either EOEntity or a subclass of EOEntity).

See Also: `className` **(EOEntity class)**

clientClassPropertiesForEntity

```
public static NSArray clientClassPropertiesForEntity(
   com.apple.yellow.eoaccess.EOEntity anEntity)
```

Returns an array containing the properties that are bound to the client-side class corresponding to *anEntity*. If no information about the client-side class's properties is available, this method returns *anEntity*'s class properties. The properties returned by this method are the attributes and relationships that are used by the client. Only these attributes and relationships will be shipped to the client.

See Also: `classProperties` (EOEntity class), `clientClassPropertyNamesForEntity`

clientClassPropertyAttributeNamesForEntity

```
public static NSArray clientClassPropertyAttributeNamesForEntity(
   com.apple.yellow.eoaccess.EOEntity anEntity)
```

Returns the names of those properties obtained using `clientClassPropertiesForEntity` that are attributes.

See Also: `clientClassPropertyNamesForEntity`, `clientClassPropertyToManyRelationshipNamesForEntity`, `clientClassPropertyToOneRelationshipNamesForEntity`

clientClassPropertyNamesForEntity

```
public static NSArray clientClassPropertyNamesForEntity(
   com.apple.yellow.eoaccess.EOEntity anEntity)
```

Returns an array containing the names of the properties that are bound to the client-side class corresponding to *anEntity*. If no information about the client-side class's properties is available, this method returns the names of *anEntity*'s class properties. The property names returned by this method are the attributes and relationships that are used by the client. Only these attributes and relationships will be shipped to the client.

See Also: `clientClassPropertiesForEntity`

clientClassPropertyToManyRelationshipNamesForEntity

```
public static NSArray clientClassPropertyToManyRelationshipNamesForEntity(
   com.apple.yellow.eoaccess.EOEntity anEntity)
```

Returns the names of those properties obtained using `clientClassPropertiesForEntity` that are to-many relationships.

See Also: `clientClassPropertyAttributeNamesForEntity`, `clientClassPropertyNamesForEntity`, `clientClassPropertyToOneRelationshipNamesForEntity`

clientClassPropertyToOneRelationshipNamesForEntity

```
public static NSArray clientClassPropertyToOneRelationshipNamesForEntity(
   com.apple.yellow.eoaccess.EOEntity anEntity)
```

Returns the names of those properties obtained using `clientClassPropertiesForEntity` that are to-one relationships.

See Also: `clientClassPropertyAttributeNamesForEntity`, `clientClassPropertyNamesForEntity`, `clientClassPropertyToManyRelationshipNamesForEntity`

clientToManyRelationshipKeysForClassDescription

```
public static NSArray clientToManyRelationshipKeysForClassDescription(
   com.apple.yellow.eocontrol.EOClassDescription aClassDescription)
```

Returns an array containing the names of those to-many relationships that are bound to the client-side class that corresponds to *aClassDescription*'s EOEntity.

See Also: `clientClassPropertyToManyRelationshipNamesForEntity`

clientToOneRelationshipKeysForClassDescription

```
public static NSArray clientToOneRelationshipKeysForClassDescription(
   com.apple.yellow.eocontrol.EOClassDescription aClassDescription)
```

Returns an array containing the names of those to-one relationships that are bound to the client-side class that corresponds to *aClassDescription*'s EOEntity.

See Also: `clientClassPropertyToOneRelationshipNamesForEntity`

EODistributionContext

Inherits from:	NSObject
Package:	com.apple.yellow.eodistribution.EODistributionContext

Class Description

An EODistributionContext object encodes data to send to the client and decodes data received from the client over the distribution channel. An EODistributionContext is also responsible for tracking the state of the server-side object graph and communicating any changes to the client, thus keeping the client and server object graphs in sync. EODistributionContext—or, if implemented, its delegate—validates remote invocations originating from client objects. The server-side EODistributionContext communicates with the EODistributedObjectStore on the client. See the EODistributionContext.Delegate interface description for more information on security and validation.

Constants

EODistributionContext defines String constants for the names of the notifications it posts. For more information, see "Notifications" (page 12).

Constructors

EODistributionContext

```
public com.apple.yellow.eodistribution.EODistributionContext(
   com.apple.yellow.webobjects.WOSession session,
   com.apple.yellow.eocontrol.EOEditingContext editingContext)
```

```
public com.apple.yellow.eodistribution.EODistributionContext(
   com.apple.yellow.webobjects.WOSession session)
```

Creates a new EODistributionContext for use within *session* and with *editingContext*, if provided, or with *session*'s default editing context otherwise.

Instance Methods

delegate

```
public Object delegate()
```

Returns the receiver's delegate.

editingContext

```
public com.apple.yellow.eocontrol.EOEditingContext editingContext()
```

Returns the receiver's editing context.

See Also: `EODistributionContext` constructor

invocationTarget

`public Object invocationTarget()`

Returns the target object to which client requests are sent for processing.

See Also: `responseToClientMessage`

responseToClientMessage

`public NSData responseToClientMessage(NSData` *`message`*`)`

Called to generate the response to a client request. The target object specified with `setInvocationTarget` is invoked with the client request, and the response returned by the target object is returned from this method.

See Also: `invocationTarget`

session

`public com.apple.yellow.webobjects.WOSession session()`

Returns the receiver's session.

See Also: `EODistributionContext` constructor

setDelegate

`public void setDelegate(Object` *`delegate`*`)`

Specifies that *`delegate`* should be used by the EODistributionContext to validate method invocations and fetches requested by the client. For more information, see the EODistributionContext.Delegate interface specification.

See Also: `delegate`

setInvocationTarget

```
public void setInvocationTarget(Object invocationTarget)
```

Specifies the target object to which client requests are sent for processing.

See Also: `responseToClientMessage`

Notifications

LoadUserDefaultsNotification

Posted whenever a distribution context receives a request for user default values from a client application. Receivers can load default values (from a database, for example) and add them to the mutable dictionary provided in the notification's userInfo.

Notification object	`this`
userInfo	An NSDictionary containing a single entry with the key "defaults" and an NSMutableDictionary as the value. The keys to the mutable subdictionary are the names of the user defaults and the corresponding values are the default values themselves.

SaveUserDefaultsNotification

Posted whenever the distribution context receives user default values from a client application. Receivers can use this notification to store the default values (in a database, for example).

Notification object	`this`
userInfo	An NSDictionary containing a single entry with the key "defaults" and another NSDictionary as the value. The keys to the mutable subdictionary are the names of the user defaults and the corresponding values are the default values themselves.

WOJavaClientApplet

Inherits from: com.apple.yellow.webobjects.WOComponent

Package: com.apple.yellow.eodistribution

Class Description

WOJavaClientApplet is the web component used by Java Client applications to create and download to the client an applet of class com.apple.client.interface.EOApplet. This component passes several parameters to the applet, including the dimensions, code / codebase, and additional EOApplication-specific parameters—such as the initial EOInterfaceController subclass name and language.

WOJavaClientApplet is able to generate the HTML required by SunSoft's Java Plug-in for Microsoft's Internet Explorer and Netscape's browsers. The plug-in is usually required for Netscape, while Internet Explorer often works without it (whether or not the plug-in is required depends on the applet's contents).

Java Client applications can be started outside of a web browser using the following command-line syntax:

```
java -classpath path_list com.apple.client.eointerface.EOApplication application_url
```

When a Java Client application is started outside of a browser, the WOJavaClientApplet is still used on the server side to determine the additional EOApplication-specific parameters. Thus the bindings listed below can still apply even in the absence of a web browser.

The following tables lists those bindings used by WOJavaClientApplet:

Binding	Description
`width`	Width of applet in the HTML page.
`height`	Height of applet in the HTML page.
`useJavaPlugin`	If this flag is YES, the WOJavaClientApplet generates HTML that causes Internet Explorer and Netscape's browsers to use SunSoft's Java Plug-in.
`archive`	Standard applet parameter.
`code`	Standard applet parameter.
`codebase`	Standard applet parameter.
`distributionContext`	The EODistributionContext used by the applet to handle requests from the client. If the WOJavaClientApplet does not have a binding for the distribution context, it instantiates one with the session's `defaultEditingContext`, sets the session as the delegate of the distribution context, and itself as the invocation target.
`interfaceControllerClassName`	The class name of the initial EOInterfaceController subclass that becomes visible when an application is launched (in the applet if launched inside a browser).
`applicationClassName`	(Objective-C only) The class name of the EOApplication subclass used for the shared application object.
`language`	The preferred language for the application.
`channelClassName`	The class name of the distribution channel to be used by the client.

Constants

WOJavaClientApplet defines the following String constants in `WOJavaClientApplet.h`. Each constant corresponds to a WOJavaClientApplet binding and is a key for use in the dictionary returned by `clientSideRequestApplicationParameters`.

Constant	Corresponding Binding
`WidthKey`	`width`
`HeightKey`	`height`
`UseJavaPluginKey`	`useJavaPlugin`
`ArchiveKey`	`archive`
`CodeKey`	`code`
`CodebaseKey`	`codebase`
`DistributionContextKey`	`distributionContext`
`InterfaceControllerClassNameKey`	`interfaceControllerClassName`
`ApplicationClassNameKey`	`applicationClassName`
`LanguageKey`	`language`
`ChannelClassNameKey`	`channelClassName`

WOJavaClientApplet defines the following additional String constants.

Constant	Description
`AllParameterNamesKey`	Used internally to collect the names of all HTML parameters passed to the client (the names of all bindings of the WOJavaClientApplet), including any additional bindings that you add to the applet.
`SessionIDKey`	Used internally to identify the session with which the server side EODistributionContext is associated.
`ComponentURLKey`	Used internally to identify the WOJavaClientApplet component on the server side which corresponds to the EOApplet on the client side.

WOJavaClientApplet also defines String constants for the names of the notifications it posts. For more information, see "Notifications" (page 18).

Instance Methods

archive

```
public String archive()
```

If the applet has a binding for `archive`, the value of that binding is returned. Otherwise, the default `archive` binding—"eojavaclient.jar"—is returned.

channelClassName

```
public String channelClassName()
```

Returns the string value bound to the `channelClassName` binding. The `channelClassName` identifies the class of the object that the client uses for a distribution channel.

See Also: `interfaceControllerClassName`

clientSideRequestApplicationParameters

`public NSDictionary clientSideRequestApplicationParameters()`

Returns a dictionary with the values of all the bindings that have been set. This method is used by EOApplication on the client to warm up a Java application started outside of a browser.

See Also: `interfaceControllerClassName`

code

`public String code()`

If the applet has a binding for `code`, the value of that binding is returned. Otherwise, the default `code` binding—“com.apple.client.eointerface.EOApplet”—is returned.

codebase

`public String codebase()`

If the applet has a binding for `codebase`, the value of that binding is returned. Otherwise, this method checks to see if the request came through a web server and, if so, returns a URL relative to `cgi-bin/WebObjects` for the resource request handler. If the request didn’t come through a web server, this method returns “/WebObjects/Java”.

distributionContext

`public EODistributionContext distributionContext()`

Returns the EODistributionContext used by this component to handle client requests.

handleClientRequest

`public Object handleClientRequest()`

Using the component’s EODistributionContext, generates a response for a client request.

See Also: `responseToClientMessage` (EODistributionContext **class)**

interfaceControllerClassName

```
public String interfaceControllerClassName()
```

Returns the value bound to `interfaceControllerClassName`.

See Also: `channelClassName`, `clientSideRequestApplicationParameters`

Notifications

DidVendComponentURLNotification

Posted after the WOJavaClientApplet vends a component URL. The notification contains:

Notification Object	The WOJavaClientApplet that vended a component URL.
Userinfo	None

WillDeallocNotification

Posted whenever the WOJavaClientApplet is about to be deallocated. The notification contains:

Notification Object	The WOJavaClientApplet that’s about to be deallocated.
Userinfo	None

EODistributionContext.Delegate

(informal interface)

Package: com.apple.yellow.eodistribution.EODistributionContext

Interface Description

If a delegate object has been set, EODistributionContext sends messages to its delegate whenever the client either requests that a method be invoked on the server or asks the server to perform a fetch. The delegate can use these methods to preempt these operations, modify their results, or simply track activity.

Given that the client can ask the server to execute any method or perform a fetch using client-supplied SQL, some measure of security is needed. By default, the EODistributionContext only allows certain method invocations and a limited set of fetches. Each of the methods in this interface should return a boolean value to indicate whether the method execution or fetch should proceed.

Instance Methods

distributionContextDidReceiveData

```
public abstract NSData distributionContextDidReceiveData(
   EODistributionContext distributionContext,
   NSData data)
```

Invoked after *distributionContext* has received data. You can use this method and it's counterpart, `distributionContextWillSendData`, to implement encryption in client server communication, encrypting in `distributionContextWillSendData` and decrypting in `distributionContextDidReceiveData`.

distributionContextShouldAllowInvocation

```
public abstract boolean distributionContextShouldAllowInvocation(
   EODistributionContext distributionContext,
   Object targetObject,
   NSSelector aSelector,
   NSArray arguments)
```

Given that the client can ask to execute any method, some measure of security is needed. By default, the distribution center prevents the invocation of any method requested by the client unless the method name is prefixed with "clientSideRequest". In order to authorize other client-requested method invocations, specify an EODistributionContext delegate (usually the session) that implements the method `distributionContextShouldAllowInvocation`. Based upon the supplied target object, method selector, and arguments array, your delegate method should return`true` if the invocation should be allowed, or`false` if it should be blocked.

See Also: `distributionContextShouldFollowKeyPath`

distributionContextShouldFetchObjectsWithFetchSpecification

```
public abstract boolean distributionContextShouldFetchObjectsWithFetchSpecification(
   EODistributionContext distributionContext,
   com.apple.yellow.eocontrol.EOFetchSpecification fetchSpecification)
```

If implemented, this delegate method is invoked when the client asks to perform a fetch by passing a fetch specification to the server. Since a fetch specification can contain arbitrary SQL it may be dangerous to allow everything. The default behavior if the delegate does not implement this method is to authorize everything but raw rows, custom SQL and lock on fetch. Based upon the supplied fetch specification, your delegate method should return `true` if the fetch should be allowed, or `false` if not. The delegate can also modify the fetch specification if needed.

distributionContextShouldFollowKeyPath

```
public abstract boolean distributionContextShouldFollowKeyPath(
   EODistributionContext distributionContext,
   String path)
```

Given that the client can ask to execute any method on any key path, some measure of security is needed. By default, the distribution center prevents the invocation of any method on any key path requested by the client unless that key path is an empty string. In order to authorize other client-requested method invocations, specify an EODistributionContext delegate (usually the session) that implements the method `distributionContextShouldFollowKeyPath`. Based upon the supplied path, your delegate method should return`true` if the key path should be followed, or `false` if not.

See Also: `distributionContextShouldAllowInvocation`

distributionContextWillSendData

```
public abstract NSData distributionContextWillSendData(
   EODistributionContext distributionContext,
   NSData data)
```

Invoked before *`distributionContext`* sends data to the client. You can use this method and it’s counterpart, `distributionContextDidReceiveData`, to implement encryption in client/server communication, encrypting in `distributionContextWillSendData` and decrypting in `distributionContextDidReceiveData`.

EODistributedDataSource

Inherits from:	EODataSource : Object
Package:	com.apple.client.eodistribution

Class Description

EODistributedDataSource is a concrete subclass of EODataSource (defined in EOControl) that fetches using an EOEditingContext as its source of objects; the editing context, in turn, forwards the fetch requests to its object store (usually an instance of EODistributedObjectStore) where it is ultimately serviced by an EODatabaseContext on the server. Objects of this class are for use with Java Client only; there is no equivalent class for Yellow Box applications. EODistributedDataSource implements all the functionality defined by EODataSource: In addition to fetching objects, it can insert and delete them (provided the entity isn't read-only). See the EODataSource class specification for more information on these topics.

EODistributedDataSource provides several methods in addition to those defined by EODataSource. The additional methods—`fetchEnabled` and `setFetchEnabled`, `fetchSpecification` and `setFetchSpecification`, and `setAuxiliaryQualifier`—are added to support enabling and disabling fetching and to support fetching with an EOFetchSpecification.

Method Types

Fetching objects

- `fetchObjects`
- `setFetchSpecification`
- `fetchSpecification`
- setAuxiliaryQualifier

Enabling fetching

- `setFetchEnabled`
- `fetchEnabled`
- `setEditingContext`

Constructors

EODistributedDataSource

```
public EODistributedDataSource(String entityName)

public EODistributedDataSource(
   EOEditingContext anEditingContext,
   String entityName)

public EODistributedDataSource(
   EOEditingContext anEditingContext,
   String entityName,
   String fetchSpecification)
```

Creates and returns a new EODistributedDataSource for the entity identified by `entityName`. If `anEditingContext` is provided, the new data source uses it as its source of objects and fetching is enabled. If it isn't provided, you must assign one with `setEditingContext`; until you do, fetching is disabled. The three-argument constructor allows you to designate a fetch specification (`fetchSpecification`) to be used by the initialized instance.

See Also: `setFetchEnabled`

Instance Methods

fetchEnabled

```
public boolean fetchEnabled()
```

Returns `true` if fetching is enabled, `false` if not.

See Also: `EODistributedDataSource` constructor, `setFetchEnabled`, `setEditingContext`

fetchObjects

```
public NSArray fetchObjects()
```

If fetching is enabled, fetches and returns objects with the receiver's fetch specification; returns `null` otherwise.

fetchSpecification

```
public EOFetchSpecification fetchSpecification()
```

Returns the receiver's fetch specification, which fetches all the objects for the receiver's entity until it is further restricted with `setFetchSpecification` or `setAuxiliaryQualifier`.

setAuxiliaryQualifier

```
public void setAuxiliaryQualifier(EOQualifier aQualifier)
```

Assigns auxiliary qualifier *`aQualifier`* to the receiver's fetch specification. This qualifier is combined with the qualifier with the fetch specification with an AND.

setEditingContext

```
public void setEditingContext(EOEditingContext anEditingContext)
```

Sets the receiver's editing context to *`anEditingContext`*. If *`anEditingContext`* is `null`, fetching is disabled.

See Also: `setFetchEnabled`

setFetchEnabled

```
public void setFetchEnabled(boolean flag)
```

Sets whether or not fetching is enabled in the receiver.

See Also: `EODistributedDataSource` constructor, `setEditingContext`

setFetchSpecification

`public void setFetchSpecification(EOFetchSpecification fetchSpec)`

Assigns *fetchSpec* to the receiver as the fetch specification to use when fetching objects.

See Also: `fetchSpecification`

EODistributedObjectStore

Inherits from:	com.apple.client.eocontrol.EOObjectStore
Implements:	NSInlineObservable
Package:	com.apple.client.eodistribution

Class Description

An EODistributedObjectStore functions as an object store on the Java client. It handles interaction with the distribution layer's channel (an EODistributionChannel object), incorporating knowledge of that channel so it can forward messages it receives from the server to its editing contexts and forward messages from its editing contexts to the server. With the channel, it represents a single connection to the server, fetching and saving objects on behalf of one or more client-side editing contexts. In this regard, an EODistributedObjectStore acts like an EODatabaseContext on the server; it differs from EODatabaseContext in that its editing contexts interact directly with it without the intervention of an object store coordinator.

EODistributedObjectStore provides several methods in addition to those defined by EOObjectStore. The invocation methods `invokeRemoteMethod` (two overloaded versions) and `invokeRemoteMethodWithKeyPath` allow you to send messages to any object on the server and receive responses from them. The methods `classDescriptionForGlobalID` and `snapshotForSourceGlobalID` return information related to enterprise objects in the distributed object store given an object's global ID.

Objects of this class are for use with Java Client only; there is no equivalent class for Yellow Box applications.

Interfaces Implemented

NSInlineObservable

`observerData`

`setObserverData`

Method Types

Initializing objects

`initializeObject`

Getting objects

`objectsWithFetchSpecification`

`objectsForSourceGlobalID`

Getting faults

`faultForGlobalID`

`arrayFaultWithSourceGlobalID`

`refaultObject`

Saving changes to objects

`saveChangesInEditingContext`

Invalidating objects

`invalidateAllObjects`

`invalidateObjectsWithGlobalIDs`

Invoking methods on the server

`invokeRemoteMethod`

`invokeRemoteMethodWithKeyPath`

Getting object data via global IDs

`classDescriptionForGlobalID`

`snapshotForSourceGlobalID`

Constructors

EODistributedObjectStore

```
public EODistributedObjectStore(EODistributionChannel aDistributionChannel)
```

Returns an EODistributedObjectStore instance initialized with a distribution channel.

Instance Methods

arrayFaultWithSourceGlobalID

```
public NSArray arrayFaultWithSourceGlobalID(
   com.apple.client.eocontrol.EOGlobalID globalID,
   String relationshipName,
   com.apple.client.eocontrol.EOEditingContext editingContext)
```

Creates a to-many fault in the editing context *editingContext* and returns the destination objects for the to-many relationship identified by *relationshipName*; *globalID* identifies the source object for the relationship (which doesn't necessarily exist in memory yet).

See Also: `faultForGlobalID`, `refaultObject`

classDescriptionForGlobalID

```
public com.apple.client.eocontrol.EOClassDescription
   classDescriptionForGlobalID(com.apple.client.eocontrol.EOGlobalID globalID)
```

Returns the class description for the enterprise object identified by *globalID*.

See Also: `snapshotForSourceGlobalID`

faultForGlobalID

```
public com.apple.client.eocontrol.EOEnterpriseObject faultForGlobalID(
   com.apple.client.eocontrol.EOGlobalID globalID,
   com.apple.client.eocontrol.EOEditingContext editingContext)
```

Creates a to-one fault from the enterprise object identified by *globalID*, registers it in *editingContext*, and returns the fault. This method could return an already existing object.

See Also: `arrayFaultWithSourceGlobalID`, `refaultObject`

initializeObject

```
public void initializeObject(
   com.apple.client.eocontrol.EOEnterpriseObject anObject,
   com.apple.client.eocontrol.EOGlobalID globalID,
   com.apple.client.eocontrol.EOEditingContext editingContext)
```

Initializes the enterprise object *anObject* with its attributes and relationships using key-value coding; the properties of *anObject* are identified and accessed using the global ID *globalID*. For properties with EONullValues, a `null` is substituted.

See Also: `classDescriptionForGlobalID`

invalidateAllObjects

```
public void invalidateAllObjects()
```

Invoked to notify the receiver that all the properties it caches are no longer valid and that they should be refaulted. Any child object stores are also notified that the objects are no longer valid. Posts InvalidatedAllObjectsInStoreNotification after removing objects from the object store.

See Also: `invalidateObjectsWithGlobalIDs`

invalidateObjectsWithGlobalIDs

```
public void invalidateObjectsWithGlobalIDs(NSArray gidArray)
```

Invoked to notify the receiver that all of the objects identified by the global IDs in `gidArray` are no longer valid. Any child object stores are also notified that the specified objects are no longer valid. After invalidating the objects, this method posts `ObjectsChangedInStoreNotification`.

invokeRemoteMethod

```
public Object invokeRemoteMethod(
   com.apple.client.eocontrol.EOEditingContext editingContext,
   com.apple.client.eocontrol.EOGlobalID globalID,
   String methodName,
   Object[] arguments)
```

```
public Object invokeRemoteMethod(
   com.apple.client.eocontrol.EOEditingContext editingContext,
   com.apple.client.eocontrol.EOGlobalID globalID,
   String methodName,
   Object[] arguments,
   boolean shouldPush)
```

Invokes the method identified by `methodName` on the server-side enterprise object identified by the editing context `editingContext` and the EOGlobalID `globalID`. The result of the invocation is returned. The four-argument method (and the five-argument method, if `shouldPush` is `true`) pushes all changes pending on the client to the server before sending the invocation; this ensures that the states of the server and client are synchronized before the method is executed. If for performance or other reasons you do not want to push pending changes to the server, use the second method with `shouldPush` set to `false`. Note that the method without the `shouldPush` argument typically originates with one of the receiver's editing contexts.

The EODistributionContext on the server by default refuses the remote invocation unless `methodName` is prefixed with "clientSideRequest" or unless its delegate (usually the session object) implements the `distributionContextShouldAllowInvocation` method to return `true`. This mechanism exists to provide security on the server.

invokeRemoteMethodWithKeyPath

```
public Object invokeRemoteMethodWithKeyPath(
   com.apple.client.eocontrol.EOEditingContext editingContext,
   String keyPath,
    String methodName,
    Object[] arguments,
    boolean shouldPush)
```

This method is similar to `invokeRemoteMethod` except for two things. The receiver of the invocation can be any object (not just an enterprise object) that can be specified with a key path (*keyPath*). The *keyPath* argument has special semantics:

- If *keyPath* is a fully qualified key path (for example, "session.editingContext") the key path is followed starting from the WOComponent that is the invocation target of the EODistributionContext.
- If *keyPath* is an empty string, the method is invoked on the WOComponent that is the invocation target of the EODistributionContext (typically a subclass of WOJavaClientApplet).
- If *keyPath* is `null`, the method is invoked on the server side EODistributionContext.

If an actual key path is specified, the EODistributionContext on the server blocks all invocations sent with this method unless *methodName* is prefixed with "clientSideRequest" or unless the EODistributionContext's delegate (on the server) implements `distributionContextShouldAllowInvocation` *and* `distributionContextShouldFetchObjectsWithFetchSpecification`. For security reasons, the delegate must authorize the invocation and the key path in these methods.

objectsForSourceGlobalID

```
public NSArray objectsForSourceGlobalID(
    com.apple.client.eocontrol.EOGlobalID globalID,
    String relationshipName,
    com.apple.client.eocontrol.EOEditingContext editingContext)
```

Returns the destination objects for the to-many relationship identified by *relationshipName*. The source object for the relationship is identified by its global ID (*globalID*). The source object and all destination objects for the relationship belong to *editingContext*. This method first looks to find the destination objects for the relationship in a client-side cache; if that cache is empty, it requests the server to send it those objects and updates the client-side cache with them.

objectsWithFetchSpecification

```
public NSArray objectsWithFetchSpecification(
   com.apple.client.eocontrol.EOFetchSpecification fetchSpecification,
   com.apple.client.eocontrol.EOEditingContext editingContext)
```

Fetches objects from the server according to the criteria specified by fetchSpecification and returns them in an array for inclusion in *editingContext*. Updates the client-side caches with the fetched enterprise objects. Throws an exception if an error occurs.

refaultObject

```
public void refaultObject(
   com.apple.client.eocontrol.EOEnterpriseObject anObject,
   com.apple.client.eocontrol.EOGlobalID globalID,
   com.apple.client.eocontrol.EOEditingContext editingContext)
```

Turns enterprise object *anObject* back into a fault (an empty enterprise object, identified by *globalID* in *editingContext*. Objects that have been inserted but not saved, or that have been deleted, shouldn't be refaulted.

See Also: `arrayFaultWithSourceGlobalID`, `faultForGlobalID`

saveChangesInEditingContext

```
public void
   saveChangesInEditingContext(com.apple.client.eocontrol.EOEditingContext editingConte
   xt)
```

Requests the server to commit changes to the enterprise objects in *editingContext*; this message is invoked by the editing context (*editingContext*). The receiver calls back to the editing context to get the updated, deleted, and inserted objects to save and commits these changes in a single transaction. Raises an exception if any error occurs.

snapshotForSourceGlobalID

```
public NSArray snapshotForSourceGlobalID(
   com.apple.client.eocontrol.EOGlobalID globalID,
   String relationshipName)
```

Returns an array of global IDs identifying the destination objects for the to-many relationship *relationshipName* having the source global ID of *globalID*. Returns `null` if the object identified by the source global ID does not currently exist in the object store or if there is no relationship with the given name.

See Also: `classDescriptionForGlobalID`

Notifications

EOGlobalID's GlobalIDChangedNotification is posted when the global ID of an object in the store changes. See the EOGlobalID documentation for more information.

InvalidatedAllObjectsInStoreNotification

This notification is posted when all objects in the object store are invalidated; see `invalidateAllObjects`.

Notification Object	`this`
userInfo Dictionary	None.

GlobalIDChangedNotification

This EOGlobalID notification is posted when the global ID of an object in the store changes.

Notification Object	`this`
userInfo Dictionary	Use the old global ID as the key to find the new global ID.

ObjectsChangedInStoreNotification

This notification is posted on when specific objects in the object store are inserted, deleted, updated, or invalidated. This can happen as a result of an update from the server.

Notification Object	`this`
userInfo Dictionary	The global IDs of inserted, deleted, updated, and invalidated objects, accessible with EOObjectStore's (respectively) InsertedKey, DeletedKey, UpdatedKey, and InvalidatedKey.

EODistributionChannel

Inherits from:	Object
Implements:	NSInlineObservable
Package:	com.apple.client.eodistribution

Class Description

EODistributionChannel is an abstract class that defines the interface for objects implementing channels for communicating data between the client and the server in a distributed Enterprise Objects application. The com.apple.client.eodistribution package includes EOHTTPChannel, a concrete subclass of EODistributionChannel that handles communication via the HTTP protocol (the most common protocol in distributed Internet applications). You can create you own subclass of EODistributionChannel if you need client-server communication based on a different protocol such as CORBA / IIOP.

An EODistributionChannel object has a **connection dictionary** that contains the values required to establish a connection on the channel, for example port, host, and URL components. You can change the connection dictionary with the `setConnectionDictionary` method.

Objects of this class are for use with Java Client only; there is no equivalent class for Yellow Box applications.

Interfaces Implemented

NSInlineObservable

`observerData`

`setObserverData`

Method Types

Getting an EODistributionChannel

`channelWithName`

Sending data on the channel

`establishConnection`

`responseToMessage`

Setting and getting the connection dictionary

`connectionDictionary`

`connectionKeys`

`setConnectionDictionary`

Accessing the delegate

`delegate`

`setDelegate`

Static Methods

channelWithName

`public static EODistributionChannel channelWithName(String` *`className`*`)`

Returns an EODistributionChannel object instantiated from the class whose name is *`className`*. Returns `null` if there is no class with that name, or if there is an instantiation, illegal-access, or security exception.

Instance Methods

connectionDictionary

`public NSDictionary connectionDictionary()`

Returns the connection dictionary used by the receiver.

connectionKeys

`public abstract NSArray connectionKeys()`

Overridden by subclasses to return the set of keys used to access the values in the connection dictionary that the channel needs to connect with the server.

delegate

`public Object delegate()`

Returns the receiver's delegate.

establishConnection

```
public abstract void establishConnection()
```

Overridden by subclasses to establish a connection with the server using a specific protocol.

See Also: `responseToMessage`

responseToMessage

```
public abstract Object responseToMessage(
   Object aMessage,
   NSCoder aCoder)
```

Overridden by subclasses to send the message *aMessage* to the server and synchronously receive a response. Before it is sent the message should be encoded using *aCoder*.

See Also: `establishConnection`

setConnectionDictionary

```
public void setConnectionDictionary(NSDictionary aDictionary)
```

Sets the connection dictionary used by the receiver to *aDictionary*.

setDelegate

```
public void setDelegate(Object delegate)
```

Sets the receiver's delegate.

EOHTTPChannel

Inherits from:	EODistributionChannel : Object
Package:	com.apple.client.eodistribution

Class Description

An EOHTTPChannel is an object that handles communication between the client and server in distributed enterprise-objects applications using the HTTP protocol. It is commonly used in WebObjects applications that employ an Enterprise Object Java client. EOHTTPChannel is concrete subclass of EODistributionChannel.

An EODistributedObjectStore manages the flow of data over the EOHTTPChannel. It sends data from the client to the server by invoking EOHTTPChannel's `responseToMessage` message, which uses an HTTP "POST" command. Communication from the server is handled through notifications "piggybacked" onto the response.

Objects of this class are for use with Java Client only; there is no equivalent class for Yellow Box applications.

Constants

EOHTTPChannel defines the following String constants as connection keys (for more information, see `connectionKeys`):

- `ApplicationURLKey`
- `ComponentURLKey`
- `SessionIDKey`
- `PageKey`

Instance Methods

connectionKeys

```
public NSArray connectionKeys()
```

Returns the keys to the connection dictionary used by the receiver. The default keys for EOHTTPChannel are the following constants:

Key	Value
`ApplicationURLKey`	The application's base URL, specifying where your application's resources are located under the web server's document root.
`ComponentURLKey`	The URL identifying a particular Java client side component.
`SessionIDKey`	The session ID for the current session.
`PageKey`	The name of the page component involved in the current transaction.

establishConnection

```
public void establishConnection()
```

Establishes a connection with the server and begins communication using the HTTP protocol. Prior to establishing the connection, the method sets the requisite host, port, and URL information using the information in the connection dictionary. Throws an NSForwardException if a native IOException or a MalformedURLException occurs; also throws an IllegalArgumentException if required information is missing from the connection dictionary.

responseToMessage

```
public Object responseToMessage(
   Object aMessage,
   NSCoder aCoder)
```

Sends the message *aMessage* from the client to the server using the HTTP “POST” command; the message is encoded before it is sent using *aCoder*. Synchronously receives, decodes, and returns the response to the message.

Throws a RuntimeException if the method is re-entered or an NSForwardException if any native exception related to socket-creation or I/O occurs.

EODistributionChannel.Delegate

(informal interface)

Package:	com.apple.client.eodistribution.EODistributionContext

Interface Description

If a delegate object has been set, EODistributionChannel sends messages to its delegate whenever the client is about to read data sent from the server or write data going to the server. These delegate methods give you the opportunity to implement encryption in client server communication.

Instance Methods

distributionChannelWillReadFromStream

```
public abstract NSData distributionContextWillReadFromStream(
   EODistributionChannel distributionChannel,
   java.io.OutputStream stream)
```

Invoked when the receiver before *distributionChannel* reads data sent from the server. You can use this method and it's counterpart, `distributionChannelWillWriteToStream`, to implement encryption in client server communication, encrypting in `distributionChannelWillWriteToStream` and decrypting in `distributionChannelWillReadFromStream`.

distributionChannelWillWriteToStream

```
public abstract NSData distributionContextWillWriteToStream(
   EODistributionChannel distributionChannel,
   java.io.OutputStream stream)
```

Invoked before *distributionChannel* writes data to send to the server. You can use this method and it's counterpart, `distributionChannelWillReadFromStream`, to implement encryption in client/server communication, encrypting in `distributionChannelWillWriteToStream` and decrypting in `distributionChannelWillReadFromStream`.

Deprecated API

This file enumerates those EODistribution classes and methods that have been deprecated and should no longer be used. Wherever possible, notes have been included to indicate what API should be used in place of the deprecated class or method.

EODistributionContext

An EODistributionContext is now strongly associated with a WOSession. Consequently, the constructor that takes only an editing context is deprecated.

EODistributionContext

```
public com.apple.yellow.eodistribution.EODistributionContext
   (com.apple.yellow.eocontrol.EOEditingContext editingContext)
```

Deprecated in Enterprise Objects Framework 4.5. Use the constructor that takes a WOSession and an EOEditingContext instead.

COLOPHON

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.