NSDistantObject

Inherits From:	NSProxy
Conforms To:	NSCoding NSObject (NSProxy)
Declared In:	Foundation/NSDistantObject.h

lass Description

NSDistantObject is a concrete subclass of NSProxy that defines proxies for objects in other applications or threads. When an NSDistantObject receives a message, in most cases it forwards the message through its NSConnection object to the real object in another application, supplying the return value to the sender of the message if one is forthcoming, and propagating any exception back to the invoker of the method that raised it.

NSDistantObject adds two useful instance methods to those defined by NSProxy. **connectionForProxy** returns the NSConnection that handles the receiver. **setProtocolForProxy:** establishes the set of methods that the real object is known to respond to, saving the network traffic required to determine the argument and return types the first time a particular selector is forwarded to the remote proxy.

There are two kinds of NSDistantObject: local proxies and remote proxies. A local proxy is created by an NSConnection the first time an object is sent to another application. It’s used by the NSConnection for bookkeeping purposes and should be considered private. The local proxy is transmitted over the network using the NSCoding protocol to create the remote proxy, which is the object that the other application uses. NSDistantObject defines methods for an NSConnection to create instance, but they’re intended only for subclasses to override—you should never invoke them directly. Use NSConnection’s **rootProxyForConnectionWithRegisteredName:host:** method, which sets up all the required state for an object-proxy pair.

dopted Protocols

NSCoding	– encodeWithCoder: – initWithCoder:

ethod Types

Creating a local proxy	+ proxyWithLocal:connection: – initWithLocal:connection:
Creating a remote proxy	+ proxyWithTarget:connection: – initWithTarget:connection:
Getting a proxy’s NSConnection	– connectionForProxy:
Setting a proxy’s Protocol	– setProtocolForProxy:

lass Methods

proxyWithLocal:connection:

+ (NSDistantObject *)**proxyWithLocal:**(id)*anObject* **connection:**(NSConnection *)*aConnection*

Returns a local proxy for *anObject* and *aConnection*, creating it if necessary. *anObject* is an **id** in the receiver’s address space. *connection* is set as the NSConnection for the returned proxy; other applications connect to it using NSConnection’s **connectionWithRegisteredName:host:** class method.

Local proxies should be considered private to their NSConnections. Only an NSConnection should use this method to create them, and your code shouldn’t retain or otherwise use local proxies.

proxyWithTarget:connection:

+ (NSDistantObject *)**proxyWithTarget:**(id)*remoteObject* **connection:**(NSConnection *)*aConnection*

Returns a remote proxy for *remoteObject* and *aConnection*, creating it if necessary. *remoteObject* is an **id** in another thread or in another application’s address space. *aConnection* is set as the NSConnection for the returned proxy; it should have been created using NSConnection’s **connectionWithRegisteredName:host:** class method.

A remote proxy can’t be used until its NSConnection’s peer has a local proxy representing *remoteObject* in the other application.

nstance Methods

connectionForProxy

– (NSConnection *)**connectionForProxy**

Returns the NSConnection used by the receiver.

initWithLocal:connection:

– (id)**initWithLocal:**(id)*anObject* **connection:**(NSConnection *)*aConnection*

Initializes a newly allocated NSDistantObject as a local proxy for *anObject*, which is an **id** in the receiver's address space. *aConnection* is set as the NSConnection for the returned proxy; other applications connect to it using NSConnection's **connectionWithRegisteredName:host:** class method. If a proxy for *anObject* and *aConnection* already exists, the receiver is released and the existing proxy is retained and returned.

Local proxies should be considered private to their NSConnections. Only an NSConnection should use this method to create them, and your code shouldn't retain or otherwise use local proxies.

This is the designated initializer for local proxies. Returns **self**.

initWithTarget:connection:

– (id)**initWithTarget:**(id)*remoteObject* **connection:**(NSConnection *)*aConnection*

Initializes a newly allocated NSDistantObject as a remote proxy for *remoteObject*, which is an **id** in another thread or in another application's address space. *aConnection* is set as the NSConnection for the returned proxy; it should have been created using NSConnection's **connectionWithRegisteredName:host:** class method. If a proxy for *remoteObject* and *aConnection* already exists, the receiver is released and the existing proxy is retained and returned.

A remote proxy can't be used until its NSConnection's peer has a local proxy representing *remoteObject* in the other application.

This is the designated initializer for remote proxies. Returns **self**.

setProtocolForProxy:

– (void)**setProtocolForProxy:**(Protocol *)*aProtocol*

Sets the methods known to be handled by the receiver to those in *aProtocol*. Setting a protocol for a remote proxy reduces network traffic needed to determine method argument and return types.

In order to encode a message's arguments for transmission over the network, the types of those arguments must be known in advance. When they're not known, the distributed objects system must send an initial message just to get those types, doubling the network traffic for every new message sent. Setting a protocol alleviates this need for the methods defined by that protocol. You can still send messages that aren't declared in *aProtocol*; in this case the initial message is sent to determine the types, and then the real message is sent.