NEXTSTEP

Title: Encoding Foundation classes with Distributed Objects

Entry Number:
Last Updated: May 8 1995
Document Revision:

Keywords: Foundation, DO, Distributed Objects, NXTransport, NSArray, NSDictionary

Question

Q: How do I encode NSObject subclasses over-the-wire with Distributed Objects since NSObject does not implement the NXTransport protocol?

Q: I have an application that uses some Foundation objects with Distributed Objects. Do any Foundation classes implement the NXTransport protocol? If not, how can I encode Foundation objects over the wire?

Answer

A:
Sending NSObjects "by reference" or "by proxy"
Sending NSObjects by reference was not supported in the foundation shipped with NEXTSTEP 3.3 and EOF 1.0. However, this is supported in the foundation included with EOF 1.1.

EOF 1.1 includes the NXAutoreleaseConnection class, that will ensure that autoreleased objects are correctly released in non-appkit servers. However, NXAutoreleaseConnection's **connectToName:onHost:** still returns a retained object. Thus, once the client is finished with the object returned by **connectToName:onHost:**, it must send **release** or **autorelease** to it.

Sending NSObjects "by value" or "by copy"
In order to send an object over-the-wire "by value" or "bycopy" you need to implement the NXTransport protocol. However, NSObject itself does not implement the NXTransport protocol, so no NSObject subclasses can implement it. As a result, NSObject subclasses cannot be encoded bycopy.

However by substituting an Object for the NSObject during the encoding process, it is possible to achieve the same result as encoding the NSObject bycopy.   When encoding an object bycopy, the DO system does the following:

- Sends the object to be encoded an **encodeRemotelyFor:freeAfterEncoding:isBycopy:** message, and keeps the return value.
- Sends that return value the **encodeUsing:** message to encode it over the wire.
- In the receiver, an instance of the class that was encoded is created with alloc/init and it is sent a **decodeUsing:** message.
- The object returned by **decodeUsing:** is passed to the receiver of the DO message.

With categories on NSObject and it's subclasses, it is possible to return a placeholder object from **encodeRemotelyFor:freeAfterEncoding:isBycopy:**, the placeholder object gets encoded over the wire, and then it can return the appropriate NSObject subclass from it's decodeUsing: implementation.   If the placeholder object is an Object subclass, it can implement the NXTransport protocol with no problem.

The NSString, NSNumber and NSData classes in Foundation already implement the NXTransport protocol in this fashion so that they can be sent bycopy with no code changes.   For example you can encode and decode these objects with the **encodeObject:** and **decodeObject:** methods as you would any other object.   For the other Foundation classes, a category must be be added to provide this functionality.   The sample code below demonstrates how to do this for NSArray and NSDictionary.   You should be able to follow this example to encode any custom NSObject subclasses bycopy over the wire.

FoundationExtensions.[hm]:FoundationExtensions.h ¬FoundationExtensions.m ¬
FoundationExtensionsPrivate.[hm]:FoundationExtensionsPrivate.h ¬FoundationExtensionsPrivate.m ¬

Note that this code encodes the objects that NSArray and NSDictionary contain using **encodeObject:** and **decodeObject:**.   You must ensure that those objects implement the NXTransport protocol, or they will not be properly encoded.

Note also that this code will *always* send NSDictionary and NSArray objects by copy, regardless of whether the bycopy keyword was used.   In fact, even when an NSMutableArray, NSMutableDictionary, or NSMutableString is sent over the wire, the receiver gets an immutable version.

Valid for:	EOF 1.0, EOF 1.1, NEXTSTEP 3.2, NEXTSTEP 3.3