NXDecoding

Adopted By: a private class

Declared In: remote/transport.h

Protocol Description

An object that implements the NXDecoding protocol is passed as the *portal* argument for the **decodeUsing:** message of the NXTransport protocol. The object implementing the **decodeUsing:** method should send the *portal* object messages from the NXDecoding protocol to decode the data required to instantiate a local copy of the encoded object.

Every method in the NXDecoding protocol corresponds to a method in the NXEncoding protocol, and is used to receive data encoded at the other end of a connection in order to move objects that adopt the NXTransport protocol. See the Distributed Objects introduction for more information.

Instance Methods

decodeBytes:count:

– **decodeBytes:**(void *)*buffer* **count:**(int)*count*

Decodes data (of size *count* bytes) into *buffer.*

decodeData:ofType:

– **decodeData:**(void *)*data* **ofType:**(const char *)*type*

Decodes a data structure, whose fields are indicated by the character string *type*, into the buffer indicated by *data*. *type* is specified with the following format characters:

Format Character	Data Type
c	char
s	short
i	int
f	float
d	double
@	id
*	char *
%	NXAtom
:	SEL
!	int; corresponding data won’t be read or written
{<type>}	struct
[<count><type>]	array

decodeMachPort:

– **decodeMachPort:**(port_t *)*portPointer*

Decodes a Mach port and returns it in the variable indicated by *portPointer.*

decodeObject

– **decodeObject**

Decodes and returns an object. The object could have been encoded with either **encodeObject:** or **encodeObjectBycopy:**.

decodeVM:count:

– **decodeVM:**(void **)*bufferPointer* **count:**(int *)*count*

Decodes memory, returning the buffer in the variable indicated by *bufferPointer* and the size in the variable pointed to by *count*.