NSObjCTypeSerializationCallBack

Adopted By: No OpenStep classes

Declared In: Foundation/NSSerialization.h

Protocol Description

An object conforms to the NSObjCTypeSerializationCallBack protocol so that it can intervene in the serialization and deserialization process. The primary purpose of this protocol is to allow for the serialization of objects and other data types that aren't directly supported by OpenStep's serialization facility. (See the NSSerializer class specification for information on serialization.)

NSMutableData declares the method that's used to begin the serialization process:

```
- (void)serializeDataAt:(const void *)data
    ofObjCType:(const char *)type
    context:(id <NSObjCTypeSerializationCallBack>)callback
```

This method can serialize all standard Objective C types (**int**, **float**, character strings, and so on) except for objects, **union**, and **void ***. If, during the serialization process, an object is encountered, the object passed as the callback argument above is asked to provide the serialization.

Suppose that the type being serialized is a structure of this description:

```
struct stockRecord {
    NSString *stockName;
    float value;
};
```

The Objective C type code for this structure is {@f}, so the serialization process begins with this message: (Assume that **theData** is the NSMutableData object that's doing the serialization and **helper** is an object that conforms to the NSObjCTypeSerializationCallBack protocol.)

```
struct stockRecord aRecord = {@"aCompany", 34.7};

[theData serializeDataAt:&aRecord ofObjCType:"{@f}" context:helper];
```

Since the first field of the structure is an unsupported type, the helper object is sent a **serializeObjectAt:ofObjCType:intoData:** message, letting it serialize the object. **helper** might implement the method in this way:

```
- (void)serializeObjectAt:(id *)objectPtr
    ofObjCType:(const char *)type
    intoData:(NSMutableData *)theMutableData
{
    NSString *nameObject;
    char *companyName

    nameObject = *objectPtr;
    companyName = [nameObject cString];

    [theData serializeDataAt:&companyName ofObjCType:@encode(typeof(companyName))
        context:nil];
}
```

The callback object is free to serialize the target object as it wishes. In this case, **helper** simply extracts the company name from the NSString object and then has that character string serialized. Once this callback method finishes executing, the original method (**serializeDataAt:ofObjCType:context:**) resumes execution and serializes the second field of the structure. Since this second field contains a supported type (**float**), the callback method is not invoked again.

Deserialization follows a similar pattern, except in this case NSData declares the central method **deserializeDataAt:ofObjCType:atCursor:context:**. The deserialization of the example structure starts with a message to the NSData object that contains the serialized data:

```
(unsigned *)cursor = 0;

[theData deserializeDataAt:&aRecord ofObjCType:"{@f}" cursor:&cursor
    context:helper];
```

(The cursor argument is a pointer to zero since we're starting at the beginning of the data in the NSData object.)

When this method is invoked, the callback object receives a **deserializeObjectAt:ofObjCType:fromData:atCursor:** message, as declared in this protocol. The callback object can then reestablish the first field of the structure. For example, **helper** might implement the method in this way:

```
- (void) deserializeObjectAt:(id *)objectPtr
    ofObjCType:(const char *)type
    fromData:(NSData *)data
    atCursor:(unsigned *)cursor
{
    char *companyName;

    [theData deserializeDataAt:&companyName ofObjCType:"*" atCursor:cursor
        context:nil];
    *objectPtr = [[NSString stringWithCString:companyName] retain];
}
```

Instance Methods

deserializeObjectAt:ofObjCType:fromData:atCursor:

– (void)**deserializeObjectAt:**(id *)*object*
ofObjCType:(const char *)*type*
fromData:(NSData *)*data*
atCursor:(unsigned *)*cursor*

The implementor of this method decodes the referenced *object* (which should always be of *type* "@") located at the *cursor* position in the *data* object. The decoded object is not autoreleased.

See also: **deserializeDataAt:ofObjCType:context:** (NSData)

serializeObjectAt:ofObjCType:intoData:

– (void)**serializeObjectAt:**(id *)*object*
ofObjCType:(const char *)*type*
intoData:(NSMutableData *)*data*

The implementor of this method encodes the referenced *object* (which should always be of *type* "@") in the *data* object.

See also: **serializeDataAt:ofObjCType:context:** (NSMutableData)