Here is yet another new format for object initializers/contents vectors. It (now) handles everything we need (until the next time we change it).

<contentsVector> ::= <objectSpecifierList> <contentsList>*

<objectSpecifierList> ::= <objectSpecifier>* <SEPARATOR_BYTE> <DATA_BYTE>*

<objectSpecifier> ::= <NOID_BYTE> <CLASS_BYTE>

<contentsList> ::= <NOID_BYTE> <objectSpecifierList>

What this means:

A <contentsVector> tells the contents of some container (you know WHICH one because you asked). It is just a list of objects, and their contents if they themselves are containers. It consists of an <objectSpecifierList> followed by zero or more <contentsList>s. The <objectSpecifierList> tells us about the objects in the container. Each <contentsList> (if there are any) tells the contents of one of the objects in the <objectSpecifierList>. How do we know how many <contentsList>s there are? We don't. We just keep consuming them sequentially until we reach the end of the message.

The <objectSpecifierList> is the information necessary to initialize some number of new objects. If consists of zero or more <objectSpecifiers> that tell WHAT objects to create, followed by a <SEPARATOR_BYTE> (a zero byte), followed by zero or more <DATA_BYTE>s that provide the information necessary to actually initialize the objects. Each <DATA_BYTE> is an arbitrary byte of data. How do we know how many <DATA_BYTE>s there are? We look at the class descriptor for the class of each object we are initializing and there will be something there that will tell us. We consume the <DATA_BYTE>s sequentially until all the objects have been initialized.

An <objectSpecifier> tells about one object that needs to be created. It consists of a <NOID_BYTE> followed by a <CLASS_BYTE>. The <NOID_BYTE> tells what Noid to assign to the new object. It can have any value except 0 (since that is the <SEPARATOR_BYTE> that separates the <objectSpecifierList> from the <DATA_BYTE>s). The <CLASS_BYTE> tells us what class of object is to be created.

A <contentsList> tells the contents of recursively contained objects. It consists of a <NOID_BYTE> followed by an <objectSpecifierList>. The <NOID_BYTE> identifies the object whose contents are being described. The <objectSpecifierList> is just as described above, and tells of the contents themselves.