Objective-C Mapping for Sequences

The Objective-C mapping uses different mappings for sequences of value types (such as sequence
>byte>) and non-value types (such as sequence
e>string>).

On this page:

- Mapping for Sequences of Value Types in Objective-C
- Mapping of Sequences of Non-Value Types in Objective-C

Mapping for Sequences of Value Types in Objective-C

The following Slice types are value types:

- Integral types (bool, byte, short, int, long)
- Floating point types (float, double)
- · Enumerated types

Sequences of these types map to a type definition. For example:

```
enum Fruit { Apple, Pear, Orange };

sequence<byte> ByteSeq;
sequence<int> IntSeq;
sequence<Fruit> FruitSeq;
```

The three Slice sequences produce the following Objective-C definitions:

```
typedef enum {
    EXApple, EXPear, EXOrange
} EXFruit;

typedef NSData EXByteSeq;
typedef NSMutableData EXMutableByteSeq;

typedef NSData EXIntSeq;
typedef NSMutableData EXMutableIntSeq;

typedef NSData EXFruitSeq;
typedef NSData EXFruitSeq;
typedef NSData EXFruitSeq;
```

As you can see, each sequence definition creates a pair of type definitions, an immutable version named <module-prefix><Slice-name>, and a
mutable version named <module-prefix>Mutable<Slice-name>. This constitutes the entire public API for sequences of value types, that is,
sequences of value types simply map to NSData or NSMutableData. The NS(Mutable)Data sequences contain an array of the corresponding
element type in their internal byte array.



We chose to map sequences of value types to NSData instead of NSArray because of the large overhead of placing each sequence element into an NSNumber container instance.

For example, here is how you could initialize a byte sequence of 1024 elements with values that are the modulo 128 of the element index in reverse order:

Objective-C int limit = 1024; EXMutableByteSeq *bs = [NSMutableData dataWithLength:limit]; ICEByte *p = (ICEByte *)[bs bytes]; while (--limit > 0) { *p++ = limit % 0x80; }

Naturally, you do not need to initialize the sequence using a loop. For example, if the data is available in a buffer, you could use the dataWithBytesNoCopy:length methods of NSData instead.

Here is one way to retrieve the bytes of the sequence:

```
Objective-C

const ICEByte* p = (const ICEByte *)[bs bytes];
const ICEByte* limitp = p + [bs length];
while (p < limitp) {
    printf("%d\n", *p++);
}</pre>
```

For sequences of types other than byte or bool, you must keep in mind that the length of the NSData array is not the same as the number of elements. The following example initializes an integer sequence with the first few primes and prints out the contents of the sequence:

```
Objective-C

const int primes[] = { 1, 2, 3, 5, 7, 9, 11, 13, 17, 19, 23 };
    EXMutableIntSeq *is = [NSMutableData dataWithBytes:primes length:sizeof(primes)];

const ICEInt *p = (const ICEInt *)[is bytes];
int limit = [is length] / sizeof(*p);
int i;
for(i = 0; i < limit; ++i) {
    printf("%d\n", p[i]);
}</pre>
```

The code to manipulate a sequence of enumerators is very similar. For portability, you should not assume a particular size for enumerators. That is, instead of relying on all enumerators having the size of, for example, an int, it is better to use sizeof(EXFruit) to ensure that you are not overstepping the bounds of the sequence.

Mapping of Sequences of Non-Value Types in Objective-C

Sequences of non-value types, such as sequences of string, structures, classes, and so on, map to mutable and immutable type definitions of NSAr ray. For example:

```
Slice

sequence<string> Page;
sequence<Page> Book;
```

This maps to:

Objective-C typedef NSArray EXPage; typedef NSMutableArray EXMutablePage; typedef NSArray EXBook; typedef NSMutableArray EXMutableBook;

You use such sequences as you would use any other NSArray in your code. For example:

This creates a book with three pages; the first two pages contain two lines each, and the third page is empty. You can print the contents of the book as follows:

Objective-C

This prints:

```
page 1, line 1: First line of page one
page 1, line 2: Second line of page one
page 2, line 1: First line of page two
page 2, line 2: Second line of page two
page 3: <empty>
```

If you have a sequence of proxies or a sequence of classes, to transmit a null proxy or class inside a sequence, you must insert an NSNull value into the NSArray. In addition, the mapping also allows you to use NSNull as the element value of an NSArray for elements of type string, structure, sequence, or dictionary. For example, instead of inserting an empty NSArray into the book sequence in the preceding example, we could also have inserted NSNull:

Objective-C

```
EXMutableBook *book = [NSMutableArray array];
[book addObject:page1];
[book addObject:page2];
[book addObject:[NSNull null]]; // Empty page
```

See Also

- Objective-C Mapping for Modules
 Objective-C Mapping for Identifiers
 Objective-C Mapping for Built-In Types
 Objective-C Mapping for Enumerations
 Objective-C Mapping for Structures
 Objective-C Mapping for Dictionaries
 Objective-C Mapping for Constants
 Objective-C Mapping for Exceptions
 Objective-C Mapping for Interfaces