IXCompareBytes(), IXCompareUnsignedBytes(), IXCompareShorts(), IXCompareUnsignedShorts(), IXCompareLongs(), IXCompareUnsignedLongs(), IXCompareFloats(), IXCompareDoubles()

SUMMARY Compare two data items as arrays of numbers and return their ordering.

DECLARED IN btree/protocols.h

SYNOPSIS int **IXCompareBytes(**const void **data1*, unsigned short *length1*, const void **data2*, unsigned short *length2*, const void **context***)**
int **IXCompareUnsignedBytes(**const void **data1*, unsigned short *length1*, const void **data2*, unsigned short *length2*, const void **context***)**
int **IXCompareShorts(**const void **data1*, unsigned short *length1*, const void **data2*, unsigned short *length2*, const void **context***)**
int **IXCompareUnsignedShorts(**const void **data1*, unsigned short *length1*, const void **data2*, unsigned short *length2*, const void **context***)**
int **IXCompareLongs(**const void **data1*, unsigned short *length1*, const void **data2*, unsigned short *length2*, const void **context***)**
int **IXCompareUnsignedLongs(**const void **data1*, unsigned short *length1*, const void **data2*, unsigned short *length2*, const void **context***)**
int **IXCompareFloats(**const void **data1*, unsigned short *length1*, const void **data2*, unsigned short *length2*, const void **context***)**
int **IXCompareDoubles(**const void **data1*, unsigned short *length1*, const void **data2*, unsigned short *length2*, const void **context***)**

DESCRIPTION Each of these functions compares two arrays of arbitrary data, *data1* and *data2* (of *length1* and *length2* bytes respectively), and returns an integer indicating their ordering. The arrays are compared as though they contained elements of the type indicated by the function name; for example, **IXCompareUnsignedLongs()** compares *data1* and *data2* as arrays of items of type **unsigned long int**. All of these functions return an integer less than, equal to, or greater than 0, according to whether *data1* is less than, equal to, or greater than *data2*.

The data in the arrays is compared serially until one element isn't equal to the other, or until either *length1* or *length2* bytes have been exhausted in the corresponding array. If two arrays are otherwise equal, the shorter is considered the lesser in value.

All of these functions match the IXComparator function type, which has the form:

typedef int **IXComparator**(const void **data1*, unsigned short *length1*,
const void **data2*, unsigned short *length2*, const void **contex*t);

Where *data1* is an array of *length1* bytes, *data2* is an array of *length2* bytes; *context* is a pointer to arbitrary data for use by the function. Only **IXFormatComparator()** (see below) makes use of a *context* argument (for that function it's called *format*). You are free to write functions matching this type definition that use *context* in any way you choose.

RETURN These functions return an integer less than 0 if *data1* is considered less than *data2*, 0 if they are considered equal, or an integer greater than 0 if *data1* is considered greater than *data2*.

SEE ALSO **IXCompareShort()**, IXBTree class, IXComparatorSetting protocol

IXCompareDouble() → See IXCompareShort()

IXCompareDoubles() → See IXCompareBytes()

IXCompareFloat() → See IXCompareShort()

IXCompareFloats() → See IXCompareBytes()

IXCompareLong() → See IXCompareShort()

IXCompareLongs() → See IXCompareBytes()

IXCompareMonocaseStrings() → See IXCompareStrings()

IXCompareShort(), IXCompareUnsignedShort(), IXCompareLong(), IXCompareUnsignedLong(), IXCompareFloat(), IXCompareDouble()

SUMMARY Compare two data items as numbers and return their ordering.

DECLARED IN btree/protocols.h

SYNOPSIS int **IXCompareShort(**const void **data1*, unsigned short *length1*, const void **data2*, unsigned short *length2*, const void **context***)**
int **IXCompareUnsignedShort(**const void **data1*, unsigned short *length1*, const void **data2*, unsigned short *length2*, const void **context***)**
int **IXCompareLong(**const void **data1*, unsigned short *length1*, const void **data2*, unsigned short *length2*, const void **context***)**
int **IXCompareUnsignedLong(**const void **data1*, unsigned short *length1*, const void **data2*, unsigned short *length2*, const void **context***)**
int **IXCompareFloat(**const void **data1*, unsigned short *length1*, const void **data2*, unsigned short *length2*, const void **context***)**
int **IXCompareDouble(**const void **data1*, unsigned short *length1*, const void **data2*, unsigned short *length2*, const void **context***)**

DESCRIPTION **Each of these functions compares two data items, pointed to by** *data1* and *data2*, as elements of the type indicated by the function name, and returns an integer indicating their ordering (*length1*, *length2* and *context* are ignored). The items are compared by pointer dereference; for example, **IXCompareFloat()** compares the values pointed to by *data1* and *data2* as type **float**. All of these functions return an integer less than, equal to, or greater than 0, depending on whether *data1* is less than, equal to, or greater than *data2*.

For more information on comparator functions, see the **IXCompareBytes()** function description.

RETURN These functions return an integer less than 0 if *data1* is considered less than *data2*, 0 if they are considered equal, or an integer greater than 0 if *data1* is considered greater than *data2*.

SEE ALSO **IXCompareBytes()**, IXBTree class, IXComparatorSetting protocol

IXCompareShorts() → See IXCompareBytes()

IXCompareStringAndUnsigneds(), IXCompareUnsignedAndStrings()

SUMMARY Compare two data items as a combinations of strings and numbers and return their ordering.

DECLARED IN btree/protocols.h

SYNOPSIS int **IXCompareStringAndUnsigneds(**const void *_data1_, unsigned short _length1_, const void *_data2_, unsigned short _length2_, const void *_context_**)**
int **IXCompareUnsignedAndStrings(**const void *_data1_, unsigned short _length1_, const void *_data2_, unsigned short _length2_, const void *_context_**)**

DESCRIPTION These functions combine the comparisons of **IXCompareStrings()** and **IXCompareUnsignedLongs()**.

IXCompareStringAndUnsigned() compares _data1_ and _data2_ as strings until the first null character is encountered. If the strings are equal up to and including the null character, the remainders are compared in the manner of **IXCompareUnsignedLongs()**. Each length argument must be the length in bytes of the string, plus 1 for the terminating null character, plus the length in bytes of the array of unsigned integers following the string.

IXCompareUnsignedAndStrings() compares the first part of _data1_ and _data2_ as a single unsigned long integer. If those are equal, the remainders are compared in the manner of **IXCompareStrings()**. Each length argument must be the length in bytes of an unsigned long integer, plus the length in bytes of the string, plus 1 for the terminating null character.

For more information on comparator functions, see the **IXCompareBytes()** function description.

RETURN These functions return an integer less than 0 if _data1_ is considered less than _data2_, 0 if they are considered equal, or an integer greater than 0 if _data1_ is considered greater than _data2_.

SEE ALSO **IXCompareBytes()**, IXBTree class, IXComparatorSetting protocol

IXCompareStrings(), IXCompareMonocaseStrings()

SUMMARY Compare two sets of data as strings and return their ordering.

DECLARED IN btree/protocols.h

SYNOPSIS int **IXCompareStrings(**const void *_data1_, unsigned short _length1_, const void *_data2_, unsigned short _length2_, const void *_context_**)**
int **IXCompareMonocaseStrings(**const void *_data1_, unsigned short _length1_, const void *_data2_, unsigned short _length2_, const void *_context_**)**

DESCRIPTION These functions compare serial arrays of strings in the NEXTSTEP character encoding. They accept either single null-terminated strings along with their lengths, or arrays of characters forming consecutive null-terminated strings (for example, ªThis is a string\0And this is another\0º) along with the total length. Only the length argument is used to determine the length of a string; a null character is treated as any other character for comparison purposes. Both of these functions return an integer less than, equal to, or greater than 0, depending on whether _data1_ is less than, equal to, or greater than _data2_.

The data in the arrays is compared serially until one element isn't equal to the other, or until either _length1_ or _length2_ bytes have been exhausted in the corresponding array. If two arrays are otherwise equal, the shorter is considered the lesser in value.

IXCompareStrings() distinguishes between uppercase and lowercase when comparing strings. **IXCompareMonocaseStrings()** disregards case distinctions in its comparison.

For more information on comparator functions, see the **IXCompareBytes()** function description.

RETURN These functions return an integer less than 0 if *data1* is considered less than *data2*, 0 if they are considered equal, or an integer greater than 0 if *data1* is considered greater than *data2*.

SEE ALSO **IXCompareBytes()**, IXBTree class, IXComparatorSetting protocol

IXCompareUnsignedBytes() → See IXCompareBytes()

IXCompareUnsignedAndStrings() → See IXCompareStringAndUnsigneds()

IXCompareUnsignedLong() → See IXCompareShort()

IXCompareUnsignedLongs() → See IXCompareBytes()

IXCompareUnsignedShort() → See IXCompareShort()

IXCompareUnsignedShorts() → See IXCompareBytes()

IXFormatComparator()

SUMMARY Compare two arrays of data based on a type encoding and return their ordering.

DECLARED IN btree/protocols.h

SYNOPSIS int **IXFormatComparator(**const void **data1*, unsigned short *length1*, const void **data2*, unsigned short *length2*, void **format***)**

DESCRIPTION **IXFormatComparator()** compares two arrays of data, determining the data type from an Objective C type encoding supplied in *format*. For example, supplying ªlº as the format indicates that *data1* and *data2* are arrays of long integers. **IXFormatComparator()** uses the other standard comparison functions as needed on the data arrays until their ordering is decided.

IXFormatComparator() currently uses only the first type declared in the format string, except for the compound formats corresponding to **IXCompareStringAndUnsigneds()** and **IXCompareUnsignedAndStrings()**, which it recognizes. Any array lengths in the format string are ignored; the actual lengths are determined from the *length1* and *length2* arguments passed to the function. If a comparison function can't be determined from the format string, **IXCompareStrings()** is used. NeXT reserves the right to interpret more than the first type declaration in future releases, as well as structure declarations, bit fields and unions.

For more information on comparator functions, see the **IXCompareBytes()** function description. For more information on comparison formats, see the IXComparisonSetting protocol specification.

RETURN **IXFormatComparator()** returns an integer less than 0 if *data1* is considered less than *data2*, 0 if they are considered equal, or an integer greater than 0 if *data1* is considered greater than *data2*.

SEE ALSO **IXCompareBytes()**, IXBTree class, IXComparisonSetting protocol

IXLockBTreeMutex(), IXUnlockBTreeMutex()

SUMMARY Lock and unlock an IXBTree for thread-safe access

DECLARED IN btree/IXBTree.h

SYNOPSIS void **IXLockBTreeMutex(**IXBTree ***aBTree*)**
void **IXUnlockBTreeMutex(**IXBTree ***aBTree*)**

DESCRIPTION These macros expand to calls to **mutex_lock()** or **mutex_unlock()** on the mutex lock instance variable of their IXBTree. They should be used to guarantee that the IXBTree isn't accessed simultaneously by two Mach threads.

SEE ALSO **mutex_lock()** (*NEXTSTEP Operating System Software*), **mutex_unlock()** (*NEXTSTEP Operating System Software*), IXBTree class

IXReadObjectFromStore() → See IXWriteRootObjectToStore()

IXUnlockBTreeMutex() → See IXLockBTreeMutex()

IXWriteRootObjectToStore(), IXReadObjectFromStore()

SUMMARY Archive or unarchive an object to or from an IXStore

DECLARED IN store/IXStoreBlock.h

SYNOPSIS unsigned int **IXWriteRootObjectToStore(**IXStore ***aStore***, unsigned int *aHandle*, id *anObject*)**
id **IXReadObjectFromStore(**IXStore ***aStore***, unsigned int *aHandle*, NXZone ***aZone*)**

DESCRIPTION **IXWriteRootObjectToStore()** archives *anObject* into the IXStore block identified by *aStore* and *aHandle*. This involves creating a memory stream, archiving the object into that stream, resizing the block to accommodate the resulting stream length, and then copying the contents of the stream into the block. Any object that implements the **write:** method using the standard archiving functions (including **NXWriteObjectReference()**) can be archived in this way.
IXReadObjectFromStore() unarchives an object from the IXStore block identified by *aStore* and *aHandle*, allocating the unarchived object from *aZone*. For further information on object archiving, see the description for the **NXReadObject()** function in the Application Kit documentation.

RETURN **IXWriteRootObjectToStore()** returns *aHandle*. **IXReadObjectFromStore()** returns the **id** of the unarchived object.

SEE ALSO **NXWriteRootObject()** (Application Kit Function), **NXReadObject()** (Application Kit Function), IXStoreBlock class.