# Chapter 4—Unicode Character Conversion

## Introduction

This section provides the reference information you need to convert your Omnis External Components to Unicode so they will run in Omnis Studio 5.0, which is a Unicode-only release. The information here is also useful for developers using Studio 4.x versions who wish to create External Components for the Unicode version of Studio 4.x.

When building Unicode components for Omnis Studio, the following pre-processor definitions should be added to the project settings: ***isunicode***, ***UNICODE*** and ***\_UNICODE***. These enable wide character versions of certain system functions and Omnis API calls.

To maintain backwards compatibility with the non-Unicode version of Omnis Studio, you should create separate targets for the Unicode-Debug and Unicode-Release versions of your components.

In this way, you can maintain a single set of source files for both Unicode and non-Unicode targets by making use of conditional-compilation statements where necessary, i.e.

###### #ifdef isunicode   // Unicode specific code here #else   // Non-Unicode specific code here #endif

In the Unicode version of Omnis Studio, all character data exchanged with external components should use the UTF-32 encoding (4 bytes per character).

There are a number of utility classes and helper functions provided by the component library and these can be found in chrbasic.he, omstring.h & omstring.c.

## Unicode Data Types

The following data types are used by the component library for handling character data.

### qchar

When *isunicode* is defined, the qchar data type is defined as unsigned long (4 bytes) and is used to contain UTF-32 data. For non-Unicode targets, qchar defaults to unsigned char.

### qoschar

When *isunicode* is defined, the qoschar data type is set to match the operating system API encoding. For Windows and Mac OS X, this is UTF-16. For Linux this is UTF-8. Thus for Windows and Mac OS X, qoschar is defined as unsigned short and for Linux, qoschar is defined as char. When *isunicode* is not defined, qoschar is defined as char.

### qbyte

The qbyte data type is always defined as unsigned char and is used for binary data and to distinguish ASCII character data from Unicode data.

## Utility Classes

### CHRconvToOs

This class converts a string of qchar data to the operating system API encoding.

#### CHRconvToOs::CHRconvToOs()

|  |
| --- |
| CHRconvToOs::CHRconvToOs(strxxx &pString) |

Creates a CHRconvToOs object from the supplied strxxx object.

#### CHRconvToOs::CHRconvToOs()

|  |
| --- |
| CHRconvToOs::CHRconvToOs(qchar \*pAdd, qlong pLen) |

Creates a CHRconvToOs object from the supplied qchar character buffer.

* pAdd point to the buffer containing UTF-32 data
* pLen is the length of the source data in characters

#### CHRconvToOs::CHRconvToOs()

|  |
| --- |
| CHRconvToOs::CHRconvToOs(qchar \*pAdd) |

Creates a CHRconvToOs object from the supplied qchar buffer. The buffer must be null-terminated.

#### CHRconvToOs::convToOs()

|  |
| --- |
| qlong CHRconvToOs::convToOs(qchar \*pAdd, qlong pLen, qoschar \*pDestBuffer) |

Converts the supplied qchar buffer to qoschars, returning the result in pDestBuffer.

* pAdd is the source buffer containing UTF-32 data
* pLen is the length of the source data in characters
* pDestBuffer is a user-allocated destination buffer, which must be large enough to accommodate the converted data.

#### CHRconvToOs::dataPtr()

|  |
| --- |
| qoschar\* CHRconvToOs::dataPtr() |

Returns a pointer to the converted data. The memory associated with this pointer is managed by the object.

#### CHRconvToOs::len()

|  |
| --- |
| qlong CHRconvToOs::len() |

Returns the length in bytes of the converted data contained inside the object.

### CHRconvFromOs

This class converts a string of characters from the operating system encoding to the Omnis internal encoding (qchars).

#### CHRconvFromOs::CHRconvFromOs()

|  |
| --- |
| CHRconvFromOs::CHRconvFromOs(qoschar \*pAdd, qlong pLen) |

Creates a CHRconvFromOs object from a buffer of qoschars.

* pLen is the length in characters of the source data

#### CHRconvFromOs::CHRconvFromOs()

|  |
| --- |
| CHRconvFromOs::CHRconvFromOs(qoschar \*pAdd) |

Creates a CHRconvFromOs object from a null-terminated string of qoschars, i.e. terminated by two consecutive null bytes when qoschar is defined as unsigned short.

#### **CHRconvFromOs::CHRconvFromOs()** Mac OS X only

|  |
| --- |
| CHRconvFromOs::CHRconvFromOs(CFStringRef pCFStringRef) |

Creates a CHRconvFromOs object from the supplied CFStringRef parameter.

#### CHRconvFromOs::convFromOs()

|  |
| --- |
| qlong CHRconvFromOs::convFromOs(qoschar \*pSrcAdd, qlong pSrcLen, qchar \*pDestAdd, qlong pDestMaxLen) |

Converts the supplied source data, writing the converted data into pDestAdd. Returns the number of characters converted.

* pSrcAdd points to the buffer containing the source data
* pSrcLen is the length of the source data in characters
* pDestAdd points to the user-allocated destination buffer which must be large enough to contain the converted data
* pDestMaxLen is the maximum length of the destination buffer in characters

#### CHRconvFromOs::pascalStringFromOs()

|  |
| --- |
| void CHRconvFromOs::pascalStringFromOs(qoschar \*pSrcAdd, qlong pSrcLen, qchar \*pDestStr, qlong pDestMaxLen) |

Converts the supplied source data, writing the converted data into pDestStr. Character position zero of the converted data contains the length in characters (0-255).

* pSrcAdd points to a buffer containing the source data
* pSrcLen is the length of the source data in characters
* pDestStr is a user allocated buffer which must be large enough to contain the converted data
* pDestStr is the maximum size of the destination buffer in character units.

#### CHRconvFromOs::dataPtr()

|  |
| --- |
| qchar\* CHRconvFromOs::dataPtr() |

Returns a pointer to the converted data, the memory for which is managed by the object.

#### CHRconvFromOs::len()

|  |
| --- |
| qlong CHRconvFromOs::len() |

Returns the length of the converted data in character units.

### CHRconvToAscii

This class converts a string of qchar data to ASCII bytes and assumes that the source data contains 7-bit ASCII compatible characters.

#### CHRconvToAscii::CHRconvToAscii()

|  |
| --- |
| CHRconvToAscii::CHRconvToAscii(strxxx &pString) |

Creates a CHRconvToAscii object from the supplied strxxx object.

#### CHRconvToAscii::dataPtr()

|  |
| --- |
| char\* CHRconvToAscii::dataPtr() |

Returns a pointer to the converted data, the memory for which is managed by the object.

#### CHRconvToAscii::len()

|  |
| --- |
| qlong CHRconvToAscii::len() |

Returns the length of the converted data.

### CHRunicode

This class provides conversion functions between different Unicode encodings.

#### CHRunicode::utf8EncodeChar()

|  |
| --- |
| qlong CHRunicode::utf8EncodeChar(qulong pChar, qbyte \*pOutUtf8, qbool pCanFatal) |

Encodes a single character as UTF-8, and returns the encoded length in bytes.

* pChar is a single UTF-32 character value
* pOutUtf8 is a user-allocated destination buffer which must be at least 4 bytes in size
* pCanFatal allows the object to generate a fatal error on conversion failure

#### CHRunicode::getUtf8EncodedChar()

|  |
| --- |
| qulong CHRunicode::getUtf8EncodedChar(qbyte \*pBuffer, qlong pInLen, qlong &pIndex, qbool pAlwaysUTF8 = qfalse) |

Gets a UTF-8 encoded character from the source buffer. Returns the converted character value as UTF-32.

* pBuffer points to the address of a UTF-8 character string
* pInLen is the length in bytes of the entire UTF-8 string
* pIndex is the byte offset from pBuffer to the start of the UTF-8 character
* pAlwaysUTF8 has no effect when isunicode is defined. Passing the value qfalse when isunicode is not defined maps the UTF-32 character to the Omnis 8 bit character set (or 0xc0 (inverted question mark) if the UTF-32 is not in the Omnis 8 bit character set)

#### CHRunicode::charToUtf8()

|  |
| --- |
| qlong CHRunicode::charToUtf8(qchar \*pInChar, qlong pInLen, qbyte \*pOutUtf8) |

Converts a string of Unicode characters to UTF-8. The output buffer length must be >= UTF8\_MAX\_BYTES\_PER\_CHAR\*pInLen bytes. Returns the encoded length.

* pInChar points to a buffer of UTF-32 characters
* pInLen is the size of the source buffer in character units
* pOutUtf8 points to a user-allocated destination buffer

#### CHRunicode::utf8ToChar()

|  |
| --- |
| qlong CHRunicode::utf8ToChar(qbyte \*pInUtf8, qlong pInLen, qchar \*pOutChar, qlong pOutBufLen = 0) |

Converts UTF-8 encoded data to Unicode (UTF-32). Returns the length of the converted data in character units.

* pInUtf8 points to the source buffer
* pInLen is the length of the source data in bytes
* pOutChar points to a user-allocated destination buffer
* pOutBufLen is the maximum size of the destination buffer in characters

#### CHRunicode::convertOmnisToUnicode()

|  |
| --- |
| void CHRunicode::convertOmnisToUnicode(qbyte \*pOmnisDataChars, qlong pLen, strxxx &pDestStr) |

Converts Omnis non-Unicode data, and stores the result in pDestStr.

* pOmnisDataChars points to a string of 8 bit data in the Omnis character set
* pLen is the length of the source data
* pDestStr is a strxxx object passed by reference

#### CHRunicode::convertOmnisToUnicode()

|  |
| --- |
| void CHRunicode::convertOmnisToUnicode(qbyte \*pOmnisDataChars, qlong pLen, handle &pDest) |

Converts Omnis non-Unicode data, and stores result in handle memory.

* pOmnisDataChars points to a string of 8 bit data in the Omnis character set
* pLen is the length of the source data
* pDestStr is a handle passed by reference

#### CHRunicode::convertOmnisToUnicode()

|  |
| --- |
| void CHRunicode::convertOmnisToUnicode(qbyte \*pOmnisDataChars, qlong pLen, qchar \*pDest, qlong pDestBufLen = 0) |

Converts Omnis non-Unicode data, and stores the result in pDest

* pOmnisDataChars points to a string of 8 bit data in the Omnis character set
* pLen is the length of the source data
* pDestStr points to a user-allocated buffer; large enough to contain the converted data
* If supplied, pDestBufLen specifies the maximum length of the destination buffer in bytes

#### CHRunicode::encodedCharactersToChar()

|  |
| --- |
| qlong CHRunicode::encodedCharactersToChar(qbool pAlwaysUtf8, qbyte \*pInEncChar, qlong pInLen, qchar \*pOutChar, qlong pOutBufLen = 0) |

Converts UTF-8/Omnis non-Unicode characters to UTF-32/qchar. Returns the length of the converted data in character units.

* pAlwaysUtf8 specifies that the source data is UTF-8 encoded data. If qfalse, it is assumed to be in the Omnis 8 bit character set
* pInEncChar points to a buffer containing the source data
* pInLen is the length in bytes of the source data
* pOutChar is a user-allocated destination buffer
* If supplied, pOutBufLen specifies the maximum length of the destination buffer in bytes

#### CHRunicode::charToEncodedCharacters()

|  |
| --- |
| qlong CHRunicode::charToEncodedCharacters(qbool pAlwaysUtf8, qchar \*pInChar, qlong pInLen, qbyte \*pOutEncChar) |

Converts qchar characters to UTF-8/Omnis characters. Returns the length in bytes of the converted data.

* pAlwaysUtf8 specifies that UTF-8 should be generated as the output; otherwise, the data is converted to the Omnis 8 bit character set
* pInChar points to a buffer containing the source data
* pInLen is the length of the source data in character units
* pOutEncChar is a user-allocated buffer large enough to contain the converted data

#### CHRunicode::setEncodingMode()

|  |
| --- |
| void CHRunicode::setEncodingMode(qbool pUtf8) |

Sets the encoding mode for encodedCharactersToChar and charToEncodedCharacters (UTF-8 or Omnis).

If qtrue, this setting overrides *pAlwaysUtf8* and specifies that conversion to/from UTF-8 is required.

#### CHRunicode::isBigEndian()

|  |
| --- |
| qbool CHRunicode::isBigEndian() |

Returns qtrue if the *ordermsb* preprocessor definition was used (i.e. if multi-byte characters are stored with the most significant byte first), qfalse otherwise.

#### CHRunicode::is7Bit()

|  |
| --- |
| qbool CHRunicode::is7Bit(qchar \*pAdd, qlong pLen) |

Returns qtrue if the source data contains entirely 7-bit data (such that UTF-8 and Omnis encodings are identical), qfalse otherwise.

* pAdd points to a buffer containing UTF-32 data
* pLen is the length of the source data in character units

#### CHRunicode::isUtf8Data()

|  |
| --- |
| qbool CHRunicode::isUtf8Data(qbyte \*pAdd, qlong pLen) |

Returns qtrue if the data satisfies the UTF-8 encoding rules. Note that this does not preclude the possibility that a non-UTF-8 string may pass this check where the source string contains extended ASCII characters and these coincide with UTF-8 encoding bytes.

* pAdd points to a buffer containing 8 bit/UTF-8 data
* pLen is the length of the source data in bytes

### CHRconvToUtf16

This class converts a string of UTF-8 data to the UTF-16 encoding.

#### CHRconvToUtf16:: CHRconvToUtf16()

|  |
| --- |
| CHRconvToUtf16::CHRconvToUtf16(qbyte \*pAdd, qlong pLen, qbool pSwap = qfalse, qbool pAddBom = qfalse) |

Creates a CHRconvToUtf16 object from the supplied source data.

* pAdd points to a buffer containing UTF-8 data
* pLen is the length of the source data in bytes
* pSwap specifies that the output byte ordering should be reversed (See CHRunicode::isBigEndian())
* pAddBom specifies that an additional Byte-Order-Marker should be placed at element zero of the converted data

#### CHRconvToUtf16::dataPtr()

|  |
| --- |
| UChar \* CHRconvToUtf16::dataPtr() |

Returns a pointer to the converted data, the memory for which is managed by the object.

#### CHRconvToUtf16::len()

|  |
| --- |
| qlong CHRconvToUtf16::len() |

Returns the length of the converted data in bytes.

### CHRconvFromUtf16

This class converts a string of UTF-16 encoded data to UTF-8

#### CHRconvFromUtf16:: CHRconvFromUtf16()

|  |
| --- |
| CHRconvFromUtf16::CHRconvFromUtf16(UChar \*pAdd, qlong pLen, qbool pSwap = qfalse) |

Creates a CHRconvFromUtf16 object from the supplied source data.

* pAdd points to a buffer containing UTF-16 encoded data
* pLen is the length of the source data in bytes
* pSwap specifies that the byte ordering of the source data is opposite to the platform default

#### CHRconvFromUtf16::dataPtr()

|  |
| --- |
| qbyte\* CHRconvFromUtf16::dataPtr() |

Returns a pointer to the converted data, the memory for which is managed by the object.

#### CHRconvFromUtf16::len()

|  |
| --- |
| qlong CHRconvFromUtf16::len() |

Returns the length of the converted data in bytes.

### CHRconvToBytes

This class converts a character buffer to a stream of bytes. For Unicode targets, the characters are encoded using UTF-8; in the non-Unicode version, the characters are unchanged.

#### CHRconvToBytes::CHRconvToBytes()

|  |
| --- |
| CHRconvToBytes::CHRconvToBytes (qchar \*pAdd, qlong pLen) |

Creates a CHRconvToBytes object from the supplied source data.

* pLen is the length of the data pointed to by pAdd in character units

#### CHRconvToBytes::CHRconvToBytes()

|  |
| --- |
| CHRconvToBytes::CHRconvToBytes (qchar \*pAdd) |

Creates a CHRconvToBytes object from the supplied source data.

* pAdd points to a null-terminated string of qchars

#### CHRconvToBytes::dataPtr()

|  |
| --- |
| qbyte \* CHRconvToBytes::dataPtr() |

Returns a pointer to the converted data, the memory for which is managed by the object.

#### CHRconvToBytes::len()

|  |
| --- |
| qlong CHRconvToBytes::len() |

Returns the length of the converted data in bytes.

#### **CHRconvToBytes::makeCanonical()** Mac OS X only

|  |
| --- |
| void CHRconvToBytes::makeCanonical() |

Makes the UTF-8 representation canonical, which is the required representation for Mac OS X file system calls. The canonical representation decomposes all composed characters (e.g. e+acute accent) into their components (e.g. the letter e, followed by acute accent symbol).

#### CHRconvToBytes::makeUtf8PascalString()

|  |
| --- |
| void CHRconvToBytes::makeUtf8PascalString(qchar \*pAdd, qlong pLen, qbyte \*pPascalString, qlong pPascalStringBufferLength) |

Converts the supplied source data to UTF-8 with a length byte at element zero , hence the length of the source data is limited to 255 characters.

* pAdd points to a buffer containing the source data
* pLen is the length of the source data in characters. 255 maximum
* pPascalString points to a user-allocated destination buffer
* pPascalStringBufferLength is the maximum size of the destination buffer in bytes

### CHRconvFromBytes

This class converts a buffer of 8 bit/UTF-8 encoded characters to qchars . For Unicode targets, the source data can be UTF-8. For non-Unicode targets, the characters are unchanged.

#### CHRconvFromBytes::CHRconvFromBytes()

|  |
| --- |
| CHRconvFromBytes::CHRconvFromBytes (qbyte \*pAdd, qlong pLen) |

Creates a CHRconvFromBytes object from the supplied source data.

* pAdd points to a buffer containing the 8 bit/UTF-8 data
* pLen is the length of the source data in bytes

#### CHRconvFromBytes::CHRconvFromBytes()

|  |
| --- |
| CHRconvFromBytes::CHRconvFromBytes (qbyte \*pAdd) |

Creates a CHRconvFromBytes object from the supplied source data.

* pAdd points to a null-terminated string of 8 bit/UTF-8 data

#### CHRconvFromBytes::dataPtr()

|  |
| --- |
| qchar \* CHRconvFromBytes::dataPtr() |

Returns a pointer to the converted data, the memory for which is managed by the object.

#### CHRconvFromBytes::len()

|  |
| --- |
| qlong CHRconvFromBytes::len() |

Returns the length of the converted data in character units.

### CHRconvFromLatin1ApiBytes

This class converts a string of Windows Latin 1 bytes to qchars.

#### CHRconvFromLatin1ApiBytes::CHRconvFromLatin1ApiBytes()

|  |
| --- |
| CHRconvFromLatin1ApiBytes::CHRconvFromLatin1ApiBytes(qbyte \*pAdd, qlong pLen) |

Creates a CHRconvFromLatin1ApiBytes object from the supplied source data.

* pAdd points to a buffer containing the Windows Latin 1 encoded data
* pLen is the length of the source data in bytes

#### CHRconvFromLatin1ApiBytes::CHRconvFromLatin1ApiBytes()

|  |
| --- |
| CHRconvFromLatin1ApiBytes::CHRconvFromLatin1ApiBytes(qbyte \*pAdd) |

Creates a CHRconvFromLatin1ApiBytes object from the supplied source data.

* pAdd points to a null terminated string of Windows Latin 1 encoded data

#### CHRconvFromLatin1ApiBytes::dataPtr()

|  |
| --- |
| qchar \* CHRconvFromLatin1ApiBytes::dataPtr() |

Returns a pointer to the converted data, the memory for which is managed by the object.

#### CHRconvFromLatin1ApiBytes::len()

|  |
| --- |
| qlong CHRconvFromLatin1ApiBytes::len() |

Returns the length of the converted data in character units.

### CHRconvToLatin1ApiBytes

This class converts a string of qchar data to the Windows/Latin1 code page.

#### CHRconvToLatin1ApiBytes::CHRconvToLatin1ApiBytes()

|  |
| --- |
| CHRconvToLatin1ApiBytes::CHRconvToLatin1ApiBytes(qchar \*pAdd, qlong pLen) |

Creates a CHRconvToLatin1ApiBytes object from the supplied source data.

* pAdd points to the source buffer containing qchar data
* pLen is the length of the source data in character units

#### CHRconvToLatin1ApiBytes::CHRconvToLatin1ApiBytes()

|  |
| --- |
| CHRconvToLatin1ApiBytes::CHRconvToLatin1ApiBytes(qchar \*pAdd) |

Creates a CHRconvToLatin1ApiBytes object from the supplied source data.

* pAdd points to a null terminated string of qchar data

#### CHRconvToLatin1ApiBytes::dataPtr()

|  |
| --- |
| qbyte \* CHRconvToLatin1ApiBytes::dataPtr() |

Returns a pointer to the converted data, the memory for which is managed by the object.

#### CHRconvToLatin1ApiBytes::len()

|  |
| --- |
| qlong CHRconvToLatin1ApiBytes::len() |

Returns the length of the converted data in bytes.

### CHRconvToEncodedCharacters

This class converts a string of qchar data to UTF-8 or Omnis 8 bit data.

#### CHRconvToEncodedCharacters::CHRconvToEncodedCharacters()

|  |
| --- |
| CHRconvToEncodedCharacters::CHRconvToEncodedCharacters(qbool pAlwaysUtf8, qchar \*pAdd, qlong pLen, csettype pSrcCset = csetOdata) |

Creates a CHRconvToEncodedCharacters object from the supplied source data.

* pAlwaysUtf8 specifies that the source data always contains Unicode characters
* pAdd points to a buffer containing rthe source data
* pLen is the length of the source data in character units
* For non-Unicode data, pSrcCset specifies the Omnis character set used for the source data. Character set constants are defined in basics.h

#### CHRconvToEncodedCharacters::dataPtr()

|  |
| --- |
| qbyte \* CHRconvToEncodedCharacters::dataPtr() |

Returns a pointer to the converted data, the memory for which is managed by the object.

#### CHRconvToEncodedCharacters::len()

|  |
| --- |
| qlong CHRconvToEncodedCharacters::len() |

Returns the length of the converted data in bytes.

#### **CHRconvToEncodedCharacters::makeCanonical()** Mac OS X only

|  |
| --- |
| void CHRconvToEncodedCharacters ::makeCanonical() |

Makes the UTF-8 representation canonical, for MacOSX file system calls. Assumes that the buffer contains UTF-8 data.

### CHRconvFromEncodedCharacters

This class converts a string of Omnis 8 bit or UTF-8 encoded data to qchars.

#### CHRconvFromEncodedCharacters::CHRconvFromEncodedCharacters()

|  |
| --- |
| CHRconvFromEncodedCharacters::CHRconvFromEncodedCharacters (qbool pAlwaysUtf8, qbyte \*pAdd, qlong pLen, csettype pDestCset = csetOdata) |

Creates a CHRconvFromEncodedCharacters from the supplied source data.

* pAlwaysUtf8 specifies that conversion from UTF-8 will definitely be required
* pAdd points to a buffer containing the source data
* pLen contains the length of the source data in bytes
* pDestCset specifies the Omnis character set to be assumed when handling ASCII data.  
  Omnis character set constants are defined in basics.h

#### CHRconvFromEncodedCharacters::CHRconvFromEncodedCharacters()

|  |
| --- |
| CHRconvFromEncodedCharacters::CHRconvFromEncodedCharacters (qbool pAlwaysUtf8, qbyte \*pAdd) |

Creates a CHRconvFromEncodedCharacters from the supplied source data.

* pAdd points to a null terminated string of UTF-8 characters

#### CHRconvFromEncodedCharacters::dataPtr()

|  |
| --- |
| qchar \* CHRconvFromEncodedCharacters ::dataPtr() |

Returns a pointer to the converted data, the memory for which is managed by the object.

#### CHRconvFromEncodedCharacters::len()

|  |
| --- |
| qlong CHRconvFromEncodedCharacters::len() |

Returns the length of the converted data in character units.

### CHRconvToOmnis

This class converts a string of qchar data to the 8 bit Omnis character set (csetOdata). No conversion is performed for non-Unicode targets.

#### CHRconvToOmnis:: CHRconvToOmnis()

|  |
| --- |
| CHRconvToOmnis::CHRconvToOmnis(qchar \*pAdd, qlong pLen) |

Creates a CHRconvToOmnis object from the supplied source data.

* pAdd points to a buffer containing the source data
* pLen contains the length of the source data in character units

#### CHRconvToOmnis::dataPtr()

|  |
| --- |
| qbyte \* CHRconvToOmnis::dataPtr() |

Returns a pointer to the converted data, the memory for which is managed by the object.

#### CHRconvToOmnis::len()

|  |
| --- |
| qlong CHRconvToOmnis::len() |

Returns the length of the converted data in bytes.

### CHRconvFromOmnis

This class converts a string of 8 bit Omnis character set data to qchars. The source data is assumed to be from the Omnis character set (csetOdata). No conversion is performed for non-Unicode targets.

#### CHRconvFromOmnis::CHRconvFromOmnis()

|  |
| --- |
| CHRconvFromOmnis::CHRconvFromOmnis(qbyte \*pAdd, qlong pLen) |

Creates a CHRconvFromOmnis object from the supplied source data.

* pAdd points to a buffer containing the source data
* pLen contains the length of the source data in bytes

#### CHRconvFromOmnis::dataPtr()

|  |
| --- |
| qchar \* CHRconvFromOmnis ::dataPtr() |

Returns a pointer to the converted data, the memory for which is managed by the object.

#### CHRconvFromOmnis::len()

|  |
| --- |
| qlong CHRconvFromOmnis::len() |

Returns the length of the converted data in character units.

### CHRconvToUniChar

This class converts from qchar to UniChar (16 bit Unicode).

#### **CHRconvToUniChar::CHRconvToUniChar()** Mac OS X only

|  |
| --- |
| CHRconvToUniChar::CHRconvToUniChar() |

Creates an empty CHRconvToUniChar object for subsequent initialisation.

#### **CHRconvToUniChar::set()** Mac OS X only

|  |
| --- |
| void CHRconvToUniChar::set(qchar \*pAdd, qlong pLen) |

Initialises the CHRconvToUniChar object from the supplied source data.

* pAdd points to a buffer containing the source data (qchars)
* pLen contains the length of the source data in character units

#### CHRconvToUniChar::CHRconvToUniChar()

|  |
| --- |
| CHRconvToUniChar::CHRconvToUniChar(qchar \*pAdd, qlong pLen) |

Creates a CHRconvToUniChar using the supplied source data. The source data must contain characters in the csetApi character set.

* pAdd points to a buffer containing the source data
* pLen contains the length of the source data in character units

#### CHRconvToUniChar::dataPtr()

|  |
| --- |
| UniChar \* CHRconvToUniChar ::dataPtr() |

Returns a pointer to the converted data, the memory for which is managed by the object.

#### CHRconvToUniChar::len()

|  |
| --- |
| qlong CHRconvToUniChar::len() |

Returns the length of the converted data in UniChar units.

### CHRconvFromCodePage

This class converts a string of 8 bit encoded character data in the specified code page to qchars. Code page constants (*preUniType …)* can be found in dmconst.he

#### CHRconvFromCodePage::CHRconvFromCodePage()

|  |
| --- |
| CHRconvFromCodePage::CHRconvFromCodePage(preconst pCodePage, qbyte \*pAdd, qlong pLen) |

Creates a CHRconvFromCodePage object from the supplied source data.

* pCodePage specifies the code page used by the source data
* pAdd points to a buffer containing rthe source data
* pLen contains the length of the source data in bytes

#### CHRconvFromCodePage::dataPtr()

|  |
| --- |
| qchar \* CHRconvFromCodePage::dataPtr() |

Returns a pointer to the converted data, the memory for which is managed by the object.

#### CHRconvFromCodePage::len()

|  |
| --- |
| qlong CHRconvFromCodePage::len() |

Returns the length of the converted data in character units.

#### CHRconvFromCodePage::codePageOk()

|  |
| --- |
| qbool CHRconvFromCodePage::codePageOk() |

Returns qtrue if the object successfully retrieved the specified code page information, qfalse if the specified code page is not supported.

#### CHRconvFromCodePage::getCodePage()

|  |
| --- |
| qushort \* CHRconvFromCodePage::getCodePage(preconst pCodePage) |

Returns a code page array of 256 unsigned shorts that are used to provide the mapping from the code page to UTF-32. Each code page has its own mapping indexed by the 8 bit data values for the code page.

### CHRconvToCodePage

This class converts a string of qchars the specified 8 bit code page. Source characters are assumed to be from the specified code page and are mapped accordingly. Any characters not present in the specified code page are mapped to ‘.’.

#### CHRconvToCodePage::CHRconvToCodePage()

|  |
| --- |
| CHRconvToCodePage::CHRconvToCodePage(preconst pCodePage, qchar \*pAdd, qlong pLen) |

Creates a CHRconvToCodePage object from the supplied source data.

* pCodePage specifies the destination code page to be assumed. See dmconst.he for a list of *preUniType…* constants.
* pAdd points to a buffer containing the source data
* pLen contains the length of the source data in character units

#### CHRconvToCodePage::dataPtr()

|  |
| --- |
| qbyte \* CHRconvToCodePage::dataPtr() |

Returns a pointer to the converted data, the memory for which is managed by the object.

#### CHRconvToCodePage::len()

|  |
| --- |
| qlong CHRconvToCodePage::len() |

Returns the length of the converted ASCII data in bytes.

#### CHRconvToCodePage::codePageOk()

|  |
| --- |
| qbool CHRconvToCodePage::codePageOk() |

Returns qtrue if the object successfully retrieved the specified code page information, qfalse if the specified code page is not supported.

#### CHRconvToCodePage::getCodePage()

|  |
| --- |
| qbyte \* CHRconvToCodePage::getCodePage(preconst pCodePage) |

Returns the reverse code page mapping table; an array which is indexed by Unicode character values. The first 4 bytes of the array (cast to a long) indicate the number of significant bytes in the array. Unicode characters past the end of the array do not exist in the code page, and are mapped as a dot.

### CHRconvFromUnicodeEncoding

This class converts a string of data from the specified encoding to the Omnis internal encoding. The encoding is specified using one of the *preUniType…* constants defined in dmconst.he

#### CHRconvFromUnicodeEncoding::CHRconvFromUnicodeEncoding()

|  |
| --- |
| CHRconvFromUnicodeEncoding::CHRconvFromUnicodeEncoding(preconst pReadEncoding, qbyte \*pData, qlong pByteLen) |

Creates a CHRconvFromUnicodeEncoding object from the supplied source data.

* pReadEncoding specifies the encoding of the source data, for example; *preUniTypeNativeCharacters*
* pData points to a buffer containing the source data (cast as qbyte \*)
* pLen specifies the length of the source data in bytes

#### CHRconvFromUnicodeEncoding::isChar()

|  |
| --- |
| qbool CHRconvFromUnicodeEncoding::isChar() |

Returns qtrue if the data after conversion is character data as opposed to binary data.

#### CHRconvFromUnicodeEncoding::charDataPtr()

|  |
| --- |
| qchar \* CHRconvFromUnicodeEncoding::charDataPtr() |

Returns a pointer to the converted data, the memory for which is managed by the object.

#### CHRconvFromUnicodeEncoding::charLen()

|  |
| --- |
| qlong CHRconvFromUnicodeEncoding::charLen() |

Returns the length of the converted data in character units.

#### CHRconvFromUnicodeEncoding::dataPtr()

|  |
| --- |
| qbyte \* CHRconvFromUnicodeEncoding::dataPtr() |

Returns a pointer to the raw converted data (cast as qbytes), the memory for which is managed by the object.

#### CHRconvFromUnicodeEncoding::len()

|  |
| --- |
| qlong CHRconvFromUnicodeEncoding::len() |

Returns the length of the converted data in bytes.

#### CHRconvFromUnicodeEncoding::getCset()

|  |
| --- |
| Csettype CHRconvFromUnicodeEncoding::getCset() |

Returns a *preUniType…* constant representing the character set used to perform the conversion.

### CHRconvToUnicodeEncoding

This class converts a string of qchars to the specified Unicode encoding. The encoding is specified using one of the *preUniType* constants defined by dmconst.he. Character data for the non-Unicode version must be in the Omnis character set (except when writing native characters or binary data).

#### CHRconvToUnicodeEncoding::CHRconvToUnicodeEncoding()

|  |
| --- |
| CHRconvToUnicodeEncoding::CHRconvToUnicodeEncoding(preconst pWriteEncoding, qbyte \*pData, qlong pByteLen, qbool pAddBom = qtrue) |

Creates a CHRconvToUnicodeEncoding object from the supplied source data.

* pWriteEncoding specifies the target encoding
* pData is a pointer to the source data (cast as qbyte \*)
* pByteLen specifies the length of the source data in bytes.
* pAddBom specifies that element zero of the output should contain a Byte-Order-Marker, used for example when writing Unicode data to external files.

#### CHRconvToUnicodeEncoding::dataPtr()

|  |
| --- |
| qbyte \* CHRconvToUnicodeEncoding::dataPtr() |

Returns a pointer to the converted data, the memory for which is managed by the object.

#### CHRconvToUnicodeEncoding::len()

|  |
| --- |
| qlong CHRconvToUnicodeEncoding::len() |

Returns the length of the converted data in bytes.

### CHRconvToUtf32FromChar

Intended for use with non-Unicode targets, this class operates on a string of qchar data, converting it to the UTF-32 encoding.

#### CHRconvToUtf32FromChar::CHRconvToUtf32FromChar()

|  |
| --- |
| CHRconvToUtf32FromChar::CHRconvToUtf32FromChar(qchar \*pData, qlong pLen, qbool pOppositeEndian, qbool pAddBom = qfalse) |

Creates a CHRconvToUtf32FromChar object from the supplied source data.

* pData is a pointer to the source data
* pLen specifies the length of the source data in character units
* pOppositeEndian specifies that the byte-endian order of the output characters should be the opposite of the platform default
* pAddBom specifies that a Byte-Order-Marker should be placed at element zero of the converted data, used for example when writing Unicode data to external files

#### CHRconvToUtf32FromChar::dataPtr()

|  |
| --- |
| U32Char \* CHRconvToUtf32FromChar::dataPtr() |

Returns a pointer to the converted UTF-32 data, the memory for which is managed by the object.

#### CHRconvToUtf32FromChar::len()

|  |
| --- |
| qlong CHRconvToUtf32FromChar::len() |

Returns the length of the converted data in charcter units.

### CHRconvFromUtf32ToChar

This class operates on a string of encoded UTF-32 data, stripping out any Byte-Order-Marker and optionally reversing the byte-endian order. Intended for use with non-Unicode targets.

#### CHRconvFromUtf32ToChar::CHRconvFromUtf32ToChar()

|  |
| --- |
| CHRconvFromUtf32ToChar ::CHRconvFromUtf32ToChar(U32Char \*pData, qlong pLen, qbool pOppositeEndian) |

Creates a CHRconvFromUtf32ToChar object for the supplied source data.

* pData specifies a pointer to the source data
* pLen specifies the length of the source data in charcter units
* pOppositeEndian specifies that the byte-order of the source data should be read in the opposite order to the platform default

#### CHRconvFromUtf32ToChar::dataPtr()

|  |
| --- |
| qchar \* CHRconvFromUtf32ToChar::dataPtr() |

Returns a pointer to the converted data, the memory for which is managed by the object.

#### CHRconvFromUtf32ToChar::len()

|  |
| --- |
| qlong CHRconvFromUtf32ToChar::len() |

Returns the length of the converted data in character units.

## Other Functions

The following functions are found in omstring.h and provide additional support for Unicode (UTF-32) character strings.

### OMstr… Functions

There are a number of Omnis string functions to mirror the standard C string functions. These operate on null-terminated strings of qchars and are prefixed to distinguish them from their ASCII counterparts. Functions include:

qulong OMstrlen(const qchar \*pString)

qchar\* OMstrcpy(qchar \*pDest, const qchar \*pSource)

qchar\* OMstrncpy(qchar \*pDest, const qchar \*pSource, qlong pCount)

qchar\* OMstrcat(qchar \*pDest, const qchar \*pSource)

qchar\* OMstrncat(qchar \*pDest, const qchar \*pSource, qlong pCount)

qbool OMstrequal(const qchar \*pString1, const qchar \*pString2)

qchar\* OMstrstr(const qchar \*pString, const qchar \*pStrCharSet)

qchar\* OMstrchr(const qchar \*pString, qchar pChar)

qchar\* OMstrrchr(const qchar \*pString, qchar pChar)

qlong OMstrcspn(const qchar \*pString, const qchar \*pStrCharSet )

qlong OMstrcmp(const qchar \*pString1, const qchar \*pString2)

qlong OMstrncmp(const qchar \*pString1, const qchar \*pString2, qlong pCount)

qlong OMstrspn(const qchar \*pString, const qchar \*pStrCharSet )

qchar\* OMstrpbrk(const qchar \*pString, const qchar \*pStrCharSet)  
qchar\* OMstrtok(OMstrtokContext \*pContext, CHR \*pStrToken, const CHR \*pStrDelimit)

The OMstrtok() function expects a *context* parameter. This has been added in order to make the function thread safe. Whareas the single-threaded C equivalent of this function is re-entrant, it assumes the same previous/partially tokenised source string when pStrToken is supplied as NULL, OMstrtok() uses pContext to store the partially tokenised source string. When called in a multi-threaded environment, this ensures that OMstrtok() is always using the correct (partially tokenised) source string. Note that the context struct must remain in scope for all calls to OMstrtok(). Example:

###### OMstrtokContext cxt;    //str255 serverInfoStr; e.g. "3.1.4" qchar \*major = OMstrtok(&cxt, &serverInfoStr[1], (qchar\*)"."); qchar \*minor = OMstrtok(&cxt,(qchar \*)NULL, (qchar\*)".");

There are also functions to convert between character strings and integers:

qchar\* OMlongToString(qchar \*pDest, qlong pLong)

qulong OMstrtoul(qchar \*pText, qchar \*\*pTextEnd, qlong pBase)

### QTEXT() Macro

This is useful for creating and supplying literal string values inside components. When \_UNICODE is defined, QTEXT() appends the L ## escape sequence onto the supplied text. This instructs the compiler to treat the resulting text as a string of qoschars. QTEXT() can be used anywhere where a qoschar\* argument is required, for example:

###### str255 myString( QTEXT("Default Value") ); //call the qoschar\* constructor

### QCHARLEN() and QOSCHARLEN() Macros

These provide a simple conversion from a supplied byte length to the corresponding qchar or qoschar character length respectively. It should be noted that they do not operate on strings or arrays of characters directly. They simply divide the supplied parameter by 4 in the case of QCHARLEN() or 2 (or 1) in the case of QOSCHARLEN().

### QBYTELEN() and QOSBYTELEN() Macros

These provide a simple conversion from a supplied character length to the corresponding UTF-32 or UTF-16/UTF-8 byte length respectively. It should be noted that they do not operate on strings or arrays of characters directly. They simply multiply the supplied parameter by 4 in the case of QCHARLEN() or 2 (or 1) in the case of QOSCHARLEN().