IBM

WebSphere MQ

Data Conversion Under WebSphere MQ

Table of Contents

Introduction

Data conversion in WebSphere MQ is a constant companion. As soon as you create a connection between two Queue Managers, data conversion has entered the picture. Regardless of whether you are connecting two ‘like’ systems (ASCII to ASCII, or EBCDIC to EBCDIC), or even if your two Queue Managers are on the same physical machine -- the WebSphere MQ data conversion routines will be invoked. In the case of a connection between two ‘like’ systems, these routines will likely decide that no work is necessary, but a WebSphere MQ trace will always show the calls to the xcsQueryCCSIDType routines whenever message data is being handled in WebSphere MQ.

Revision information

Revised June 2008: Corrections and expansion.

Revised October 2009: Corrections.

Acronyms and terms used in Data Conversion

Here are a few of the words and acronyms which may be used in this and other documents you may read about data conversion. They are 'informal' definitions

Table 1. Acronyms and terms

code page	A number which IBM and others assign to a particular assignment of character shapes (glyphs) to binary codes.
character set	A number used to identify the set of characters represented, without reference to the codes used for the characters. One character set can be encoded into more than one code page.
CCSID	Coded Character Set IDentifier. A number. An IBM term which extends the meaning of code page to include other code information. Defined formally in the CDRA (Coded Character Representation Architecture) documents from IBM (see also "Coded Character Set Identifier (CCSID)").
code set	UNIX term equivalent to code page. On UNIX systems this is a character string.
SBCS	Single Byte Character Set. A code page with up to 256 characters which can be represented by a single 8 bit byte.
DBCS	Double Byte Character Set. A code page with up to 65536 characters represented by two 8 bit bytes. Also used to mean anything related to languages which use DBCS or MBCS code pages. Note that, in practice, most languages which use DBCS also support an SBCS character set to represent Latin characters and have differing ways of switching between SBCS and DBCS in a character string.
MBCS	Multiple Byte Character Set. Some encodings used for large character set languages can use 2, 3, or 4 bytes for each character. MBCS is the general term used to cover all non-SBCS character sets, but DBCS is often used instead.
Mixed	A CCSID or Codepage which supports an SBCS subset and a DBCS or MBCS subset. The CCSID uses for WebSphere MQ queue manager data must be 'mixed' or SBCS to be able to code object name characters as single bytes.
EBCDIC	Extended Binary Coded Decimal Interchange Code: Used by IBM host systems (running z/OS or VM operating systems) or IBM iSeries using the OS/400 operating system.
ASCII	American Standard Code for Information Interchange. The code used as the basis for most codes apart from EBCDIC.
USASCII	The US English version of ASCII.
euc	Extended unix code. MBCS encoding used on UNIX platforms. These include: eucJP (Japanese), eucKR (Korean), eucTW (Traditional Chinese, used in Taiwan), and eucCN (simplified Chinese, used in the PRC).

SOSI	Shift Out / Shift In. The state dependent method used in EBCDIC SBCS / DBCS code pages to indicate the DBCS characters.
UNICODE	A single 2 byte code page which defines most of the characters used by the languages now used throughout the world. See UCS-2, UTF-8, UTF-16 and UTF-32 for encodings.
UTF-8	An encoding of UNICODE which is variable in length from 1 to 3 bytes and has the property that ASCII characters 00-7F(hex) are single bytes.
UTF-16	An encoding of UNICODE which uses a 16 bit integer to represent up to 64 thousand characters, and a further million using two 16-bit characters from the 'surrogate' range. Note that WMQ, at present, supports the UCS-2 subset of UTF-16 only.
UTF-32	An encoding of UNICODE which uses a 32 bit integer. Note that WMQ does not, at present, support UTF-32.
UCS-2	An encoding of UNICODE which uses a 16 bit integer to represent each character, now being replaced by UTF-16. UCS-2 is the subset of UTF-16 without surrogate pairs.
ISO8859	A series of ASCII based standards which define encodings for SBCS character sets used by Latin, Hebrew, Arabic, and Cyrillic based languages. ISO8859-1 is for Western Europe. These are supported on most UNIX boxes and are the basis from which some of the Microsoft Windows codepages (1250-1256) were developed.
ISO10646	The ISO universal codeset. 4 bytes per character are defined, but essentially UNICODE is a 2 byte subset.

The Pieces in the Data Conversion Puzzle

There are two pieces to the data conversion puzzle.

Coded Character Set Identifier (CCSID)

The Coded Character Set ID (CCSID), also called a Code Page, is a number assigned to a particular method of representing data. A code page will assign character and non-character data to hexadecimal values ranging from '00' to 'FF'; or '0000' to 'FFFF' in Double Byte Character Set (DBCS) code pages, which are used for alphabets that cannot be represented by 256 positions (e.g. Kanji). CCSID conversion tables, which are used to convert from one CCSID to another, generally convert character data and control characters. See Appendix A, "Control code conversions" for more details.

Some well known CCSID's:

- CCSID 437 -- this is an ASCII code page, used mainly under OS/2, DOS, and Microsoft Windows console (OEM) windows.
- CCSID 850 -- this is another common ASCII code page, used mainly on under OS/2, DOS, and Microsoft Windows console (OEM) windows used on PCs in Europe.
- CCSID 819 -- this is the ISO 8859-1 standard Western European ASCII code page. Used by most UNIX locales. The related CCSID with Euro support is 923.
- CCSID 500 -- This is an EBCDIC code page, used mainly on z/OS. It is known as the 'International' codepage. The related CCSID with Euro support is 1148.
- CCSID 037 -- This is another popular EBCDIC code page, used on OS/400 and VSE/ESA. It is the 'US English' codepage. The related CCSID with Euro support is 1140.
- CCSID 1252 -- this is the windows native codepage used on Windows 95 NT, and later. It is a superset of CCSID 819: It has extra graphic characters but does not support the ISO control characters in hex 80-9F locations.

Encoding

Encoding is generally taken to mean the method that this platform uses to represent numeric data [1]. There are two general types:

- 'LittleEndian', used by Intel processors (e.g Windows, Linux on Intel). In this encoding, the least significant digits appear in lower memory locations, e.g. the number 437 would be represented (in hex) as X'BF01'.

- 'BigEndian'. The most significant digits are in lower memory locations. e.g. the number 437 would be represented (in hex) as X'01BF'.

As you can see from the above explanations, even numeric data may need conversion; even if passed between different platforms which might use the same CCSID. Here is a list of some of the platforms used by WebSphere MQ (client and server) and the encoding scheme used on that platform:

Table 2. Encoding scheme by platform

Platform	Encoding Scheme
zSeries	BigEndian
AIX	BigEndian
iSeries	BigEndian
HP-UX	BigEndian
Windows	LittleEndian
SINIX	BigEndian
Sun Solaris (on SPARC processors)	BigEndian
Sun Solaris (on INTEL processors)	LittleEndian
Linux (Intel)	LittleEndian
Linux (zSeries)	BigEndian
NonStop Kernel (NSK)	BigEndian
OVMS Alpha	BigEndian
Open VMS VAX	BigEndian
Tru64 Unix	BigEndian

Footnotes:

1. For the pedants, numeric in this document means integer numeric. WebSphere MQ does not, at present, convert floating point formats.

What Gets Converted, and How

WebSphere MQ needs to perform data conversion for two different purposes.

- For Channel Communication -- When a channel between two WebSphere MQ Queue Managers is started, the two Queue Managers will negotiate which CCSID and encoding protocol they will use for channel communications. This CCSID and encoding scheme is then used for encoding the WebSphere MQ TSH (This is the Transmission Segment Header -- a WebSphere MQ control block that you may see passed back and forth between Queue Managers if you look at a WebSphere MQ trace) and the Message Descriptor. The Message Descriptor is created and populated by your application when you do the MQPUT, but WebSphere MQ needs to be able to understand what it contains, regardless of the platform that it was created on, thus it will be converted automatically by WebSphere MQ regardless of whether the user portion of the message is converted or not.
- The User Portion of the Message -- The CCSID and encoding scheme of the user portion of the message are contained in the Message Descriptor, and were set at MQPUT time. This part of the message will not be automatically converted by WebSphere MQ without some work on the part of the user. This will be discussed in more detail below.

The Message Descriptor

The Message Descriptor is not entirely character data, so some of it will be converted as character data, some as numeric, and some is passed unconverted. The conversion of the MD is as follows:

Table 3. Message Descriptor conversion

Field Name	Field Length in bytes	Offset	Converted as
StrucId	4	‘00’x	Character
Version	4	‘04’x	Numeric
Report	4	‘08’x	Numeric
MsgType	4	‘0C’x	Numeric
Expiry	4	‘10’x	Numeric
Feedback	4	‘14’x	Numeric
Encoding	4	‘18’x	Numeric
CodedCharSetId	4	‘1C’x	Numeric
Format	8	‘20’x	Character
Priority	4	‘28’x	Numeric
Persistence	4	‘2C’x	Numeric
MsgId	24 (‘18’x)	‘30’x	Byte
CorrelId	24 (‘18’x)	‘48’x	Byte
BackoutCount	4	‘60’x	Numeric

ReplyToQ	48 (‘30’x)	‘64’x	Character
ReplyToQMgr	48 (‘30’x)	‘94’x	Character
UserIdentifier	12 (‘0C’x)	‘C4’x	Character
AccountingToken	32 (‘20’x)	‘D0’x	Byte
ApplIdentityData	32 (‘20’x)	‘F0’x	Character
PutApplType	4	‘110’x	Numeric
PutApplName	28 (‘1C’x)	‘114’x	Character
PutDate	8	‘130’x	Character
PutTime	8	‘138’x	Character
ApplOriginData	4	‘140’x	Character
GroupId	24 (‘18’x)	‘144’x	Byte
MsgSeqNumber	4	‘15C’x	Numeric
Offset	4	‘160’x	Numeric
MsgFlags	4	‘164’x	Numeric
OrginalLength	4	‘168’x	Numeric

Note: MsgId, CorrelId, Accounting Token and GroupId are NOT considered to be character data, and as such WILL NOT be converted as character data. Many applications put character data into these fields, but if the message is being exchanged between ASCII and EBCDIC platforms, and the application on the other side needs to be able to read these fields, then it is the responsibility of the application (or a user exit) to convert these fields. WebSphere MQ puts no restriction on the format of the data in these fields, and indeed, if you allow WebSphere MQ to create the MsgId field, it will NOT be strictly character data.

The User portion of the message

The user portion of a WebSphere MQ message may or may not be converted for you by WebSphere MQ. It may be that you would not want to have the message converted, because it consists entirely of numeric data (although you still need to take into account the differences in *encoding* between different platforms). It may be that only certain parts of the message need to be converted, because the message consists of mixed numeric and character data. It may be that the entire message can be safely converted, because it consists of entirely character data. Let’s look at each of these scenarios in turn.

Common Procedures when doing the MQPUT

Regardless of whether the message is character, numeric, or a mixture of both, in the application that is doing the initial MQPUT of the message, you should:

- Set MQMD.CodedCharSetId to MQCCSI_Q_MGR. This will ensure that WebSphere MQ will pick up the default CCSID of the Queue Manager you are doing the MQPUT from. The reason for this is that the CodedCharSetId field in the Message Descriptor is both an input and an output field. This means that if the application is using the same MQMD structure for both MQPUTs and MQGETs, WebSphere MQ will be putting the CCSID of every message that it MQGETs into this field. If one of these messages was not using the same CCSID as that of the Queue Manager, and you did not convert it, then this

field will have been changed to an unanticipated value when the time comes to do the next MQPUT. Even if you specified that you wished to convert the message on the MQGET, but the MQGET had an unexpected return code (such as MQRC_TRUNCATED_MSG_ACCEPTED) then the message will remain unconverted, and the CodedCharSetId field will be set to the CCSID of the unconverted message.

Note: In a client application, MQCCSI_Q_MGR refers to the CCSID of the application locale, not the CCSID of the remote queue manager. In all cases, when setting MQMD.CodedCharSetId to MQCCSI_Q_MGR, it is the responsibility of the application to ensure that the implied CCSID correctly represents the string data in the message.

- Set MQMD.Encoding to MQENC_NATIVE. This will ensure that WebSphere MQ picks up the correct encoding for the platform you are running on.

The message is entirely numeric data

If the message is entirely numeric data, *and* you know that the numeric encoding schemes of the two platforms you are passing the message between are the same, then you need to do nothing special for data conversion. If you suspect that the numeric data encoding schemes of the two platforms may be different, or if you wish to plan for a future when this might be the case, you should use the section below on mixed character and numeric data as a guide, since this will work equally well for completely numeric data.

The message is entirely character data

If the message you are passing is entirely character data then it is very easy to have WebSphere MQ do all of the work for you. In the application that is doing the MQPUT of the message, you should set MQMD.Format to MQFMT_STRING. MQFMT_STRING is a constant that equates to the string value “MQSTR “. This value in the Format field will let WebSphere MQ know that the message consists entirely of character string data.

The message is mixed numeric and character data

If the message is mixed numeric and character data, then you will need to supply a conversion user exit for WebSphere MQ to use to convert the data. WebSphere MQ provides skeleton structures and utilities to help you create these user exits. You should refer to the WebSphere MQ Application Programming Guide for complete information on this subject, but here is an outline of the steps you need to take to get this done:

1. Decide on a name for your message that will fit in the MQMD.Format field in the message descriptor (8 characters). You will need to make sure that you always fill in MQMD.Format with this name whenever you send this message.
2. Create a C-structure that represents the structure of your message, e.g.

```
typedef struct _mymsg {
        short a;
        char b[10];
        long c;
        short d;
        char e;
```

```
} Mymsg;
```

3. Run this structure through the crtmqcvx utility (for Unix and Windows) or CVTMQMDTA (iSeries) to create a code fragment for your data conversion exit.
3. There is a common sample skeleton data conversion exit called amqsvfc0.c. Now replace the commented out section in this sample with the code fragment generated above. Make sure you also put a prototype declaration for the call at the top of the sample (after the #include statements for the WebSphere MQ supplied headers is a good spot).
3. Decide on an entry point name for your code (usually EntryPointxxxxxxxx -- where 'xxxxxxxx' is the MQMD.Format name you decided on), and make sure you replace the dummy entry point name in the MQENTRY and MQDATACONVEXIT entries in the sample. Make sure you put this entry point name in the EXPORTS section of your .DEF file as well.
4. Compile your exit as a DLL (for Windows), a shared library (for UNIX), or an (OBJTYPE *PGM) (for OS/400).
5. Make sure the resulting output is in your PATH (for Windows), a link to /usr/lib or equivalent (for Unix), or the library list for the WebSphere MQ job (OS/400).

Potential problems to be aware of

- The WebSphere MQ macros, as provided, expect all structures to be packed and the C-language on most platforms will tend to put padding characters in structures between character and numeric data, if the boundaries are not on a word boundary. To overcome this:
    - on WebSphere MQ for z/OS, declare the data structure as:
      `_Packed struc_name xxxx`
    - on WebSphere MQ for AIX, specify the following compiler option:
      `-qalign=packed`
    - on WebSphere MQ for Windows, specify the following as a directive in your code:

```
#pragma _Packed        or
#pragma pack(1)
```

- Avoid the use of the INT and LONG data types, since the length of these depend on the architecture and operating system. Use MQLONG or MQSHORT instead.
- On UNIX platforms which support threading and non-threading or multiple different threading models, you should compile your exit for each of the models. If you don't build all versions you may find that that the exit is not found if it is called by different applications or by the WebSphere MQ channel code if you have specified CONVERT (YES) on your SENDER channel definition.

Common Procedures to ensure conversion

Regardless of the content of the message, if conversion is required, you should:

- In the application doing the MQGET of the message, set MQMD.CodedCharSetId to 0 to pick up the default CCSID of the QMGR. This will ensure that WebSphere MQ knows the correct CCSID to convert to.
- In the application that is doing the MQGET of the message, make sure you specify MQGMO_CONVERT as one of the MQGMO.Options. This field has many options that should be either added or bitwise OR'ed together if you need to specify more than one.
- It is recommended that you always do the data conversion on the MQGET of the message, since the message may take many 'hops' across different platforms on its way to its destination, and doing data conversion on each 'hop' is a waste of resources.
- Some of the early Version 1 and 2 products did not support data conversion. If you specify CONVERT(YES) on the SENDER channel that sends the data to its final destination, WebSphere MQ will convert the data into the CCSID of the queue manager on the target system. If you are using a conversion user exit to convert numeric or mixed numeric and character data, and you are doing the conversion by specifying CONVERT (YES) on a SENDER channel definition, then your user exit will need to reside on the system where this SENDER channel is defined. Using this option is not recommended.

Where to Find the Conversion Tables

The conversion tables are in different directories, depending on the platform. In some cases, such as WebSphere MQ for iSeries, you cannot add conversion tables -- the operating system provides all of the conversion tables you're going to get. Most of the Unix platforms provide a method for adding conversion tables; all it takes is adding the binary conversion table to a certain directory on the box.

On WebSphere MQ for AIX

Under AIX, the conversion tables go into (and existing conversion tables can be found in) the `/usr/lib/nls/loc/iconvTable` directory.

The converters use codeset names on AIX. These are usually IBM-xxxx where xxxx is the CCSID number, but there are exceptions. See Appendix D, "How CCSIDs relate to code set names" for some of these differences.

WebSphere MQ for AIX also uses the 'uconv' method for supporting new DBCS and some SBCS conversions. The tables in `/usr/lib/nls/loc/iconvTable` are only used for SBCS. These can be generated using the AIX genxlt command. The uconv method tables are more complex and use the uconvdef command. If you think you need to create new conversion methods using this method look at the articles in the AIX info-explorer documentation carefully. To determine if conversion is supported by the uconv method check the following:

In `/usr/lib/nls/loc/uconvTable` (Note uconvTable not iconvTable) there should be two files, one for the from-codeset and one for the to-codeset.

In `/usr/lib/nls/loc/iconv` there should be a link from file `from-codeset_to-codeset` to file `/usr/lib/nls/loc/iconv/Universal_UCS_Conv`

On WebSphere MQ for HP-UX

The converters use codeset names on HP-UX. These are usually cpxxxx where xxxx is the CCSID number, but there are exceptions. See Appendix D, "How CCSIDs relate to code set names" for some of these differences.

On HP-UX the conversion tables go into (and existing conversion tables can be found in) the `/usr/lib/nls/iconv/table directory.`

In the directory `/usr/lib/nls/iconv` is a file called `iconv.config` which contains aliases for character set names. If you look in this file after WebSphere MQ is installed you will see that extra entries have been made for the additional support provided by the product.

On WebSphere MQ for Sun Solaris and Linux

The conversion tables go into (and existing conversion tables can be found in) the `/opt/mqm/lib/iconv` directory.

The conversion code used is similar to Windows.

The tables on Sun Solaris are named by joining together the hexadecimal versions of the codepage numbers, so conversion from 500 to 850 will be in table 01F40352.tbl. In the conversion table directory file readme.ccs describes most of the shipped tables.

On WebSphere MQ for Windows

The conversion tables go into (and existing conversion tables can be found in) the `c:\mqm\CONV\TABLE` directory where `c:\mqm` is the disk and directory in which the user installed WebSphere MQ (c:\mqm is the default).

The tables on Windows are named by joining together the hexadecimal versions of the codepage numbers, so conversion from 500 to 850 will be in table 01F40352.tbl. In the conversion table directory file readme.ccs describes most of the shipped tables.

Testing Conversion

On WebSphere MQ for AIX and HP the `iconv` command can be used to exercise any of the installed conversion tables. Entering the command

```
iconv -f fromcodeset -t tocodeset fromfile > tofile
```

where *`fromcodeset`* is the codeset of the file *`fromfile`*, *`tocodeset`* is the codeset you want *`tofile`* to be in.

Hints and Tips

Due to the nature of 'known problems', this section of the document should be considered subject to change. You should visit the WebSphere MQ Family Product Support page if you suspect a conversion problem that is not documented here.

- On the HP-UX operating system the NLS feature is an optional installable feature that MUST be installed for data conversion to work. You should make sure that this feature is installed before you even install WebSphere MQ, since WebSphere MQ will place some code page tables in the NLS created directories.
- The HP-UX operating system default code page for Western European locales is 1051 (ROMAN8). You are recommended to change the codepage to iso8859-1 (CCSID 819) whenever you create a WebSphere MQ Queue Manager. To do this:
    - Enter 'locale -a' to find out which "ISO88591" locale is supported/installed on your system. The Quick Beginnings for WebSphere MQ for HP-UX **Chapter 10. "Code sets supported on WebSphere MQ for HP-UX"** describes the various locales for the CCSID of 819.
    - BEFORE you issue crtmqm to create your Queue Manager, issue: export LC_CTYPE=en_US.ISO88591 (or xx_YY.ISO88591 for whatever was returned from LOCALE -a)

      Once the QMGR is created, it will always use the LC_CTYPE which it was created under.
- The Application Program Reference manual contains the tables showing what conversion support is provided on each platform.
- If you have a problem with the conversion of some characters to or from EBCDIC, (especially the exclamation point), make sure that the z/OS queue manager CCSID matches the codepage the user is expecting the data in. The default CCSID of 500 has many characters with different hex values to those often used in the codepage which is being used. For a full list of these differences see Appendix F, "How do EBCDIC Latin 1 codepages differ". If this happens get the z/OS queue manager CCSID changed. See "How to set the Queue Manager CCSID on WebSphere MQ for z/OS" for instructions.
- On Unix platforms you may have to compile two versions of each conversion exit depending on the threading options you program uses. If your new format is MYFORMAT, on AIX and HP-UX the name of the extra version is MYFORMAT_r, and on Solaris it is MYFORMAT_d. See the chapter on Data-conversion exits in the Application Programming Guide for details on how to compile these extra exits. If you install the DCE option it is a good idea to compile both variants as the mover code will use DCE which requires the extra exit if you use the convert on send option. A symptom of this problem is receiving MQRC_FORMAT_ERROR (2110, x'83E'), or a message saying that the _r or _d version of the exit could not be found.
- Conversion of PCF data: If the CCSID in MD of a message containing PCF data is the same as the requesting CCSID in the MD of the MQGET structure, the WebSphere MQ workstation products do not convert the data. If individual PCF strings are in different CCSIDs they will not be converted. To force the scanning of all the PCF

structures for different CCSIDs use the value MQCCSI_EMBEDDED as the message CCSID when the message is put on the queue.

- On AIX, if you get the occasional report of a conversion problem when displaying messages from channels, and your queue manager is running in a non-Latin locale (for example Japanese), this may be due to the listener process started by `inetd` running in the 'C' locale. Work arounds which will stop this are:

  Set default data conversion (see "Default data conversion"), or modify the program called by inetd to call instead a shell script. The shell script should set the locale environment variables and then call the WebSphere MQ listener program.

- On Linux, Solaris and Windows, conversion tables are provided by WebSphere MQ. On AIX, HP and iSeries, a combination of WebSphere MQ supplied and operating system supplied conversion tables are used. Conversion tables supplied by WebSphere MQ tend provide round-trip integrity, i.e. when a piece of data is converted from one CCSID to another and back again, the original data is recovered. Round-trip tables assign a different mapping to every character. If a character only exists in the *from* CCSID and not in the *to* CCSID, an arbitrary but unique mapping is assigned to ensure that the conversion process can round trip.

  There are other ways of constructing conversion tables. One way is to map all "unknown" characters that exist in the *from* CCSID but not in the *to* CCSID to a substitute character. Such tables do not provide round-trip integrity, i.e. when a piece of data is converted from one CCSID to another and back, the original data may not be recovered. Conversion tables supplied with the operating system are often of this type.

  If you require specific handling of "unknown" characters, you may wish to perform the data conversion within your application using conversion tables and algorithms that satisfy your requirements.

Tracing (Windows, UNIX and iSeries)

When looking at trace, a return code of xecX_W_INCOMPATIBLE_CCSIDS x'10806111' or decimal 0276848913 from the function xcsCCSIDCompatible is NOT an error. It indicates that the two CCSIDs passed into the function are valid but that data conversion is required even for object names. The code should then proceed to call the function to perform this conversion.

When the conversion table is not found: The CCSIDs, and code set names used are shown. During initialization the trace will show if default conversion is selected, and, if it is, what codepages are selected.

Supporting multiple languages

WebSphere MQ can help developers of true multilingual applications but the challenges extend beyond message passing. I will not discuss the problems of GUI design and data entry as this is usually handled by a client application. Assuming that somewhere in your application is a data store, probably a data base, you need to decide if the data here is to be stored. You could use a common encoding, for example UNICODE. See "WebSphere MQ and UNICODE." for more information on using MQ and UNICODE. Alternatively, you may decide to tag data and retain different encodings for each language. You could store the CCSID with each piece of data for example.

Getting messages to flow between queue managers which have CCSIDs set to different languages can be a problem. One solution applicable to some platforms is default data conversion: see "Default data conversion". Another way is to use the change queue manager CCSID function to change all the queue managers to use CCSIDs for a single language. You can still send messages in different CCSIDs by coding the CCSID in you sending application in the MQPUT MD field.

If you want to use data conversion in you MQGET you can have a problem deciding which CCSID to convert into if you do not know the language of the message. For example, if you were sending Korean and German messages, in ASCII to an iSeries machine and wanted to convert these to EBCDIC if you cannot use a single CCSID in EBCDIC which can represent the accented characters of German and the multibyte characters of Korean. One suggestion is to use a separate queue for each language, so you can code the appropriate CCSID in the MQGET MD.

WebSphere MQ and UNICODE

What is UNICODE?

The UNICODE standards define a list of about 40000 characters which contain most of the characters used by the majority of people in the world today. This includes East and Western Europe, Arabic, Hebrew, Russian and the characters used in Mainland China, Taiwan, Korean and Japan. There are various ways these characters can be given binary values, which I will call 'encoding'. The basic encoding is to use two bytes (16 bits) to represent each character. This is the number you will find in the UNICODE standard: for details I use the (large) book The Unicode Standard, Version 3.0 **Publisher:** Addison-Wesley Professional; Bk&CD Rom edition (February 16, 2000) **ISBN:** 0201616335. A more recent edition is *The Unicode Standard, Version 4.0* (Boston, MA, Addison-Wesley, 2003. ISBN 0-321-18578-1). The UNICODE web site is also useful: http://www.unicode.org/

This encoding is called 'UCS-2' or 'UTF-16'. An alternative way of encoding the same number is one which is variable in length, and uses one to three bytes. This is called 'UTF-8'. There are other encodings such as 'UTF-7' and 'UTF-32', but these are not supported by WebSphere MQ.

Note: UCS-2 is a fixed-width encoding of 2 bytes per character. UTF-16 is an extension to UCS-2 that includes additional characters represented as surrogate pairs (4 bytes per character). Support for UTF-16 and UTF-8 in WebSphere MQ is limited to those Unicode characters that may be encoded in UCS-2.

WebSphere MQ support for UNICODE

All WebSphere MQ products will pass messages in any encoding (including any UNICODE encoding) as data, but not all can convert messages from UNICODE to other platform specific encodings. The WebSphere MQ Application Programming Reference manual has a data conversion appendix which contains some information on the platforms which support UNICODE data conversion.

The message header structures on WebSphere MQ cannot contain data encoded in UCS-2 as they are fixed length and assume one byte per character. As string data in the message header must be in the CCSID of the queue manager, UCS-2 cannot be used as the queue manager CCSID. On EBCDIC platforms, it is also not possible to use UTF-8 as the queue manager CCSID because UTF-8 is based on ASCII. If you want to pass user data in UNICODE you must always specify a UNICODE CCSID in the message descriptor (md) in an MQPUT call. Similarly, if you are using data conversion by specifying MQGMO_CONVERT in the get message options on an MQGET, you must specify a UNICODE CCSID in the md in an MQGET. You should not set the channel conversion option to YES if you are using UNICODE as this conversion will always be to the queue manager CCSID of the receiving queue manager, which cannot be UCS-2 and UTF-8 is not supported as a queue manager CCSID on all platforms.

UCS-2 and UTF-8 - which should you use?

UCS-2

Character processing UCS-2 is simpler as all characters are 16 bit integers (2 bytes), but note that basic C string functions will not work as UCS-2 data can contain many embedded NULLs. Also note that UCS-2 data is integer data so you have to be careful about the representation: On most Intel platforms this is 'little-endian' and on iSeries, zSeries and many UNIXes this is 'big-endian'.

UTF-8

UTF-8 character processing is more complex as each character can take one to three bytes to represent. UTF-8 has the useful property that the basic ASCII characters in the range x'20' to x'7E' take one byte. Also all the control character in the range x'00'-x'1F' also only take one byte. This ensures there are no unexpected NULLs in the data and makes it 'file safe'. Also, if most of your data is basic ASCII the number of bytes used is less than for UCS-2. However some characters will take more space in UTF-8 than in UCS-2. The transformation between UCS-2 and UTF-8 is described in the UNICODE standard book. WebSphere MQ can convert from UCS-2 to UTF-8.

UNICODE CCSIDs

The UCS-2 CCSIDs are 1200, 13488, and 17584. Strictly 13488 is the CCSID for UNICODE V2.0 and 17584 the CCSID for UNICODE V2.1 (which adds a few characters including the euro). WebSphere MQ will treat these as identical. The UTF-8 CCSID is 1208.

Designing UNICODE applications for WebSphere MQ

I would strongly advise a pilot application to check that all the support you need is in place. There are many ways you could deploy UNICODE to solve the multilingual problem: Here are a few.

1. Convert all you message data to/from UNICODE in your client and use WebSphere MQ as a transport layer. All your server processing is done in UNICODE. This does not use the WebSphere MQ conversion services and is particularly applicable to clients where there are good platform converters between the UNICODE and platform codepages (for example AIX or NT).
2. Convert all messages to/from UNICODE in the server and use WebSphere MQ as a transport layer. All your server processing is done in UNICODE. This does not use the WebSphere MQ conversion services and is particularly applicable to servers where there are good platform converters between the UNICODE and codepages.
3. Use WebSphere MQ to perform the conversion. This requires the queue managers at each end to be able to perform conversion to and from UNICODE.

GB18030

GB18030 is the name of the latest Simplified Chinese code standard. WebSphere MQ V5.3 or later supports conversion between GB18030 and Unicode. This support is for the ‘phase one’ subset of GB18030. Characters in GB18030 can use one, two or four bytes. Those defined in phase one can be represented in Unicode UCS-2 in two bytes. WebSphere MQ does not support the conversion of GB18030 characters which are encoded in Unicode using surrogates which require 4 bytes in the UTF-16 encoding.

Default data conversion

Ever had the problem of connecting a US English Queue Manager to a Japanese Queue Manager? Has someone wanted to set up a global network with English, Greek, Cyrillic (Russian), and Turkish machines? Usually this will fail with the channel not starting, and a "conversion not supported" message. As explained above, for any information to flow from one machine to another, data must be converted to the receiving queue manager's CCSID and encoding. In general there are no inter-language conversion tables available. How would you represent a Kanji (Japanese) character in a Latin character set? However, the data in headers which is required to process a message, and which flows between channels when they negotiate their connection, is in very restricted character set: usually A-Z, a-z, 0-9 and a few special characters. To convert this data all that is required is to know if it is ASCII or EBCDIC, then a single table can be used to perform the conversion, in theory (there are limitations: see below).

On the WebSphere MQ products (AIX, Windows, Sun Solaris, HP-UX and Linux), this is called **default data conversion**. This is enabled by defining a default ASCII CCSID and a default EBCDIC CCSID. If the platform does not have a conversion table from CCSID **A** to CCSID **B** but CCSIDs **A** and **B** are known to the queue manager, (unless it is a very new one they will be), and the defaults are defined then the following will happen:

- If the CCSIDs **A** and **B** are both ASCII or both EBCDIC, the data is passed on unconverted.
- If one of the CCSIDs is ASCII and the other EBCDIC, then the default values will be used to perform the conversion. Obviously this means that you should use an ASCII/ EBCDIC pair which is supported on your platform.

The use of **default data conversion** is intended to enable data to be transferred between queue managers whose CCSIDs represent CCSIDs for different character sets.

Enabling default data conversion

In the *ccsid.tbl* file un-comment the two lines which define the EBCDIC and ASCII default values. The suggested defaults are 500 and 850, but you can change these, making sure conversion between them is supported. This change will be picked up on the next MQCONN.

Limitations

Default data conversion is not intended for user data conversion as it may not do the conversion you expected. Conversion exits do not use default conversion without special modification to the exit code. The options parameter in the MQXCNVC calls in the user exit must include the value DCC_DEFAULT_CONVERSION. This will require the programmer of the exit to manually make this change to the user exit code, as the conversion code generating command **crtmqcvx** does not include this value in the options parameter. To include the value the programmer should "OR" the value DCC_DEFAULT_CONVERSION with the other options set in the code.

If **default data conversion** is set it **will** be used in internal conversions including, for

example, the conversion of messages with a message format of MQFMT_STRING.

Be careful if using default conversion with EBCDIC Japanese CCSIDs 290, 930 and 5026. The codes used for lower case Latin characters in these CCSIDs are different to any other EBCDIC CCSID. If you use CCSID 500 as your EBCDIC default the lower case characters converted using default conversion will be converted as defined by CCSID 500. If you want your default to use 290/930/5026 for EBCDIC change the EBCDIC default value to 930 or 5026 and the ASCII value to a CCSID which can be used for conversion to 930 or 5026, eg: 932, 94, 954, or 5050 which are ASCII or euc Japanese CCSIDs.

Another alternative when using Japanese EBCDIC codes is to make sure all the object names used for WebSphere MQ objects use only uppercase Latin characters.

Appendix A. Control code conversions

Conversion of control characters between EBCDIC and ASCII can cause problems. Some platforms make an attempt to convert them but there is a problem here, ASCII has only 32 code points for control characters while EBCDIC assigns 64, so 32 of the EBCDIC control characters will be converted to character values in ASCII. The following is a list of control characters which will, in general, be preserved during data conversion.

Table 4. Control code conversion.

Name	EBCDIC hex code	ASCII hex code	Name	EBCDIC hex code	ASCII hex code
NUL - Null	00	00	EM - End Message	19	19
SOH - Start Of Heading	01	01	FS – File Separator	1C	1C
STX - Start Of Text	02	02	IGS - Interchange Group Separator	1D	1D
ETX - End Of Heading	03	03	IRS - Interchange Record Separator	1E	1E
HT - Horizontal Tab	05	09	IUS/ITB - Interchange Unit Separator/ Intermediate Transmission Block	1F	1F
VT - Vertical Tab	0B	0B	LF - Line Feed	25	0A
FF - Form Feed	0C	0C	ETB - End of Transmission Block	26	17
CR - Carriage Return	0D	0D	ESC - Escape	27	1B
SO - Shift Out	0E	0E	ENQ - Enquiry	2D	05
SI - Shift In	0F	0F	ACK - Acknowledge	2E	06
DLE - Data Link Escape	10	10	BEL - Bell	2F	07
DC1 - Device Control 1	11	11	SYN - Synchronous Idle	32	16
DC2 - Device Control 2	12	12	EOT – End of Transmission	37	04
DC3 - Device Control 3	13	13	DC4 - Device Control 4	3C	14
BS - BackSpace	16	08	NAK - Negative Acknowledge	3D	15
CAN - Cancel	18	18	SUB – Substitute	3F	1A

Note that this table does not include the NL (hex 15) character. There is no newline character

in most ASCII CCSIDs. Most WebSphere MQ products convert this to ASCII line feed (hex 0A). See Appendix C, "How the non-z/OS platforms treat EBCDIC newline" for a detailed discussion on Newline!

Appendix B. CCSID.TBL

There has been some confusion on the use of this file. Internally WebSphere MQ needs to know various bits of information about a CCSID. This includes which conversion table to use and if it is EBCDIC or ASCII, Single byte or Multibyte.

WebSphere MQ for AIX, HP-UX

All of the well known CCSIDs have table entries internally for this information. Only if a CCSID error message is issued (eg amq6050 or amq6053) should you consider adding an entry in the ccsid.tbl file. If you are adding a new conversion table you will, in general, NOT have to add anything to the ccsid.tbl file as the CCSIDs will be known to WebSphere MQ, but the particular conversion table for the pair of CCSIDs may not have been provided.

If you think you need an entry it would be a good idea to contact WebSphere MQ service to confirm this is needed and get the correct entry to put in the table. Putting the wrong data in the table could cause further conversion errors.

On the WebSphere MQ V5 products (AIX, Windows, Sun Solaris, Linux and HP-UX), the table can be used to enable default data conversion: see "Default data conversion" for details.

WebSphere MQ for Windows, Linux, Sun Solaris

The Windows, Linux and Sun Solaris ccsid.tbl contains all the entries used by the product.

Appendix C. How the non-z/OS platforms treat EBCDIC newline

There is no perfect answer on what to do with the EBCDIC NL character. On PC ASCII (eg codepage 850) there is no control code assigned to NL, so the tables used would convert this character (when converting from EBCDIC to ASCII) to the codepoint for a character in codepage 850 which is NOT present in the EBCDIC codepage.

On ISO based codepages (eg ISO 8859-1 or Roman8) as used on HP, there is a control code assigned (hex 85). On some versions of HP, the HP conversion even does this if the codeset is a multibyte codeset, and this created invalid or unexpected DBCS characters, as hex 85 is part of some of the defined DBCS character range.

One solution would be to leave the ISO conversions unchanged on HP, but fix all the others. This would leave the problem of understanding what a New line character in WebSphere MQ is. Customer would also have the same problem when writing applications to run on different codesets or platforms. If you want your code to run on the different PC ASCII codes your newline detection would have to be aware of all the variations of conversion tables.

To fix this, WebSphere MQ for AIX, Windows, Sun Solaris, Linux and HP-UX always convert the EBCDIC NL to Line feed before using any of the conversion tables supplied, giving a consistent behavior across all platforms and codesets.

The behaviour of WebSphere MQ's conversion of NL can be modified by setting the ConvEBCDICNewline stanza in the MQS.INI file on UNIX and iSeries platforms. On Windows systems, you modify configuration information using the properties pages for WebSphere MQ, accessed from the WebSphere MQ Services snap-in.

The values supported are:

NL_TO_LF	EBCDIC NL is converted to ASCII LF. This is the default behaviour, and is how all V5.0 products behave.
TABLE	EBCDIC NL is converted to what ever value is specified in the supplied conversion tables.
ISO	EBCDIC NL is converted to what ever value is specified in the supplied conversion tables if the source is an ISO CCSID otherwise the conversion is the same as NL_TO_LF.

You should only set the ConvEBCDICNewline stanza if you are sure you want the behavior specified. It is really only of use when converting from EBCDIC to an ISO version of ASCII which includes a NL character in its control values. In this case you will get a round trip conversion for the NL character when converting from EBCDIC⇨ASCII⇨EBCDIC.

However, if you set the value to TABLE and convert to other ASCII CCSIDs you will find the NL character will be converted to many different values in ASCII. The values will be found to vary from one platform to another and between different ASCII CCSIDs on the same platform.

Appendix D. How CCSIDs relate to code set names

On many platforms, for example some UNIX, we use the iconv conversion method. This requires the use of codeset names to describe the data conversions. Here is some information on how to convert a CCSID to a code set name. In many cases the conversion table name is made up of the codeset names of the conversion.

AIX, HP-UX and DEC-OVMS

Table 5. **Codeset names and CCSIDs**

CCSIDs	Description	Codeset name		
		AIX	HP-UX	OVMS
819	Latin Western European	ISO8859-1	iso88591	iso8859-1
912	Latin Eastern European	ISO8859-2	iso88592	iso8859-2
915	Cyrillic	ISO8859-5	iso88595	iso8859-5
1089	Arabic	ISO8859-6	iso88596	iso8859-6
813	Greek	ISO8859-7	iso88597	iso8859-7
916	Hebrew	ISO8859-8	iso88598	iso8859-8
920	Turkish	ISO8859-9	iso88599	iso8859-9
950	Traditional Chinese	big5	big5	big5
954 5050 33722	Japanese	IBM-eucJP	eucJP	eucJP, deckanji, sdeckanji
970	Korean	IBM-eucKR	eucKR	deckorean
964	Traditional Chinese	IBM-eucTW	eucTW	eucTW, dechanyu
1383	Simplified Chinese	IBM-eucCN	eucCN	eucCN
1051	HP-UX Latin	IBM-1051	roman8	IBM1051
285	UK English EBCDIC	IBM-285	engle	IBM285
297	French EBCDIC	IBM-297	frene	IBM297
285	German EBCDIC	IBM-273	germe	IBM273
xxxx	usual default name	IBM-*xxxx*	cp*xxxx*	IBM*xxxx*

Appendix E. How to set the Queue Manager CCSIDs

The Queue Manager CCSID is important as it defines the default CCSID for messages being created using **MQPUT** and is the default CCSID which will be used for conversion when messages are retrieved using **MQGET**. If the channel option CONVERT(YES) is set the conversion is made into the receiving Queue Manager's CCSID.

How to set the Queue Manager CCSID on WebSphere MQ for z/OS

For z/OS change the Queue Manger CCSID from its default value of 500. To change the queue manager CCSID on z/OS use CSQ6SYSP macro setting the QMCCSID parameter. See the WebSphere MQ for z/OS System Setup Guide for more information.

Non z/OS platforms

Displaying the Queue Manager CCSID

To display the queue manager CCSID use the `DIS QMGR` command from within runmqsc.

Setting the Queue Manager CCSID on WebSphere MQ for UNIX products

The CCSID is set when the queue manager is created. The CCSID used is the CCSID for the codeset of the locale of the user running the **crtmqm** command. You can change the Queue Manager CCSID using the mqsc qmgr alter command, as well as by deleting the queue manager. You must still stop the queue manager and any other WebSphere MQ processes (for example the command server) after changing the CCSID, and then restart them otherwise there is a possibility that different processes could be using old and new CCSIDs.

Examples of CCSIDs set for different locales

```
On HP
export LANG=en_US.iso88591
 uses the codeset iso88591
 and will set a CCSID of 819
export LANG=en_US.roman8
 uses the codeset roman8
 and will set a CCSID of 1051
export LANG=C (this is the default locale)
 uses the codeset roman8
 and will set a CCSID of 1051

On AIX
export LANG=en_US
 uses the codeset ISO8859-1
```

and will set a CCSID of 819

```
export LANG=EN_US.UTF-8
 uses the codeset UTF-8
```

and will set a CCSID of 1208

```
export LANG=C (this is the default locale)
 uses the codeset ISO8859-1
 and will set a CCSID of 819
```

Setting the Queue Manager CCSID on WebSphere MQ for Windows

The CCSID is set when the queue manager is created. The CCSID used is the CCSID for the codepage of the console of the user running the **crtmqm** command. You can change the Queue Manager CCSID using the mqsc qmgr alter command, as well as by deleting the queue manager. You must still stop the queue manager and any other WebSphere MQ processes (for example the command server) after changing the CCSID, and then restart them otherwise there is a possibility that different processes could be using old and new CCSIDs.

The console codepage can be displayed and changed using the **chcp** command.

For information on which CCSID to choose for single byte languages, see Appendix G, "Windows and OEM codepages".

Note: Language specifics.

The CCSID used for Japanese Windows codepage is 932, but this can be changed by modifying an entry in the ccsid.tbl file to use 943. The CCSID 932 is used by IBM for the codepage used on IBM-PCs and is slightly different in definition to the 932 codepage defined by Microsoft for Windows. IBM has defined a CCSID 943 which matches the Microsoft definition.

The CCSID used for Simplified Chinese Windows is 1381, even though the Windows codepage is set to 936, as the IBM definition of CCSID 1381 is closer to the Microsoft codepage 936 than the IBM CCSID 936. This can be changed by modifying an entry in the ccsid.tbl file to use 1386. The CCSID 1386 (GBK) is defined by IBM to more closely match the Microsoft definition of 936.

The CCSID used for Korean Windows codepage is 949, but this can be changed by modifying an entry in the ccsid.tbl file to use 1363. The CCSID 949 is used by IBM for the codepage used on IBM-PCs and is slightly different in definition to the 949 codepage defined by Microsoft for Windows. IBM has defined a CCSID 1363 which matches the Microsoft definition.

Setting the Queue Manager CCSID on WebSphere MQ for iSeries

The CCSID is fixed when the queue manager is created. The queue manager CCSID cannot be changed. The CCSID used for the queue manager is the value of system value QCCSID. If this is set to 65535 the job default CCSID is used. To change the CCSID the queue manager must be deleted, the value of QCCSID changed and the queue manager re-created.

Note: The job default CCSID is set by the iSeries system to a valid value, and is derived form the job language and country values and cannot be changed directly by the users.

Displaying iSeries CCSID values

To display the value of QCCSID enter `dspsysval QCCSID`

To display the job default CCSID, enter `dspjob`
Select job definition attributes
Look for the value of Default Coded Character Set Identifier which is near the end of the list on screen three.

Setting the Queue Manager CCSID on MQSeries for HP OpenVMS

The CCSID is set when the queue manager is created. The CCSID used is the CCSID for the codeset of the locale of the user running the **crtmqm** command. The queue manager CCSID cannot be changed. To change the CCSID the queue manager must be deleted, the locale changed, and the queue manager re-created.

OpenVMS locales are provided by the OpenVMS operating system, The search for the locale file uses the `SYS$I18N_LOCALE logical.` To set your locale to use all the language specific settings define a logical called `SYS$LC_ALL or LC_ALL` at the system, group or job level and set this to the value of the locale you want to use. If you only want to select the actual locale category used by WebSphere MQ to set the CCSID, define and set logicals `SYS$LC_CTYPE or LC_CTYPE.`

Examples of the CCSID used for each locale value can be found in Appendix "codeset support on MQSeries for Digital OpenVMS" in the "MQSeries for Digital OpenVMS System Management Guide" publication.

To list you current locale use the **`LOCALE SHOW`** command. To list all the available locales use the **`LOCALE SHOW PUBLIC`** command. If the list does not include the locale you wish to use look on the OpenVMS binaries CD for the VMS I18N directory where you will find the installation kit for additional locales, etc. When you install this kit you can choose which territory to install. The territories supported include European plus US, Chinese, Japanese, Korean and Thai.

You can use the **`LOCALE LOAD`** command to load the locale into shared memory, which will improve the performance when using locales.

Appendix F. How do EBCDIC Latin 1 codepages differ

Almost every major country in Western Europe has its own EBCDIC codepage. This reflects the fact that these codepages were standardized when machines could only print or display 64 or 96 characters. Each country chose its favourite characters to be within this subset. When the EBCDIC codepages were defined the 192 characters in all the Latin 1 related codepages are included but the codepoints are different. A common problem reported is that conversion works except for one or two characters. This may be due to the user using on EBCDIC codepage to enter and display data, but using a different (usually codepage 500 as its the z/OS default) to do the conversion.

In the table below the differences between the common codepages are listed. The codepages listed are

37	USA, Canada
273	Germany, Austria
277	Denmark, Norway
278	Finland, Sweden
280	Italy
284	Spain, Latin America
285	United Kingdom
297	France
500	International
871	Iceland
1047	Latin 1, Open Systems (code page used by z/OS C compiler).

Table 6. EBCDIC table differences

Character (name)	Hex code used by the character in this codepage.										
	USA	Ger.	**D/N.**	F/S.	It.	Sp.	UK	Fr.	**Int**	Ice.	**Lat1**
	37	273	277	278	280	284	285	297	500	871	1047
[(left bracket)	**BA**	**63**	**9E**	**B5**	**90**	4A	**B1**	**90**	4A	**AE**	**AD**
] (right bracket)	**BB**	**FC**	**9F**	**9F**	**51**	5A	**BB**	**B5**	5A	**9E**	**BD**
{ (left brace)	C0	**44**	**9C**	**44**	**44**	C0	C0	**51**	C0	**8E**	C0
} (right brace)	D0	**DC**	**47**	**47**	**54**	D0	D0	**54**	D0	**9C**	D0
~ (tilde)	A1	**59**	**DC**	**DC**	**58**	**BD**	**BC**	**BD**	A1	**CC**	A1
@ (at sign)	7C	**B5**	**80**	**EC**	**B5**	7C	7C	**44**	7C	**AC**	7C
¦ (split vertical bar)	6A	**CC**	**70**	**CC**	**CD**	**49**	6A	**DD**	6A	6A	6A
# (number sign) (hash) (pound)	7B	7B	**4A**	**63**	**B1**	**69**	7B	**B1**	7B	7B	7B
` (grave)	79	79	79	**51**	**DD**	79	79	**A0**	79	**8C**	79
^ (hat) (circumflex)	**B0**	5F	5F	5F	5F	**BA**	**BA**	5F	5F	**EC**	5F
§ (section)	B5	**7C**	B5	**4A**	**7C**	B5	B5	**5A**	B5	B5	B5
! (exclamation point)	**5A**	4F	4F	4F	4F	**BB**	**5A**	4F	4F	4F	**5A**
$ (dollar)	5B	5B	**67**	**67**	5B	5B	**4A**	5B	5B	5B	5B
¬ (logical not)	**5F**	BA	BA	BA	BA	**5F**	**5F**	BA	BA	BA	**B0**
¨ (diaresis) (umlaut)	BD	BD	BD	BD	BD	**A1**	BD	**A1**	BD	BD	**BB**
£ (pound sterling)	B1	B1	B1	B1	**7B**	B1	**5B**	**7B**	B1	B1	B1
¢ (cent)	**4A**	B0	B0	B0	B0	B0	B0	B0	B0	B0	**4A**
| (vertical bar) (logical or)	**4F**	BB	BB	BB	BB	**4F**	**4F**	BB	BB	BB	**4F**
¤ (international currency) (sputnik)	9F	9F	**5A**	**5A**	9F	9F	9F	9F	9F	9F	9F
µ (mu)	A0	A0	A0	A0	A0	A0	A0	**79**	A0	A0	A0
ß (beta)	59	**A1**	59	59	59	59	59	59	59	59	59
\ (back slash)	E0	**EC**	E0	**71**	**48**	E0	E0	**48**	E0	**BE**	E0
° (degree)	90	90	90	90	**4A**	90	90	**4A**	90	90	90
¯ (overline)	BC	BC	BC	BC	BC	BC	**A1**	BC	BC	BC	BC
´ (acute)	BE	BE	BE	BE	BE	BE	BE	BE	BE	**E0**	BE
ä (a umlaut)	43	**C0**	43	**C0**	43	43	43	43	43	43	43
à (a grave)	44	44	44	44	**C0**	44	44	**7C**	44	44	44
å (a overcircle)	47	47	**D0**	**D0**	47	47	47	47	47	47	47
Ä (A umlaut)	63	**4A**	63	**7B**	63	63	63	63	63	63	63

Å (A overcircle)	67	67	**5B**	**5B**	67	67	67	67	67	67	67
æ (ae ligature)	9C	9C	**C0**	9C	9C	9C	9C	9C	9C	**D0**	9C
Æ (AE ligature)	9E	9E	**7B**	9E	9E	9E	9E	9E	9E	**5A**	9E
ç (c cedilla)	48	48	48	48	**E0**	48	48	**E0**	48	48	48
ð (eth)	8C	8C	8C	8C	8C	8C	8C	8C	8C	**79**	8C
Ð (Eth) (D stroke)	AC	AC	AC	AC	AC	AC	AC	AC	AC	**7C**	AC
é (e acute)	51	51	51	**79**	**5A**	51	51	**C0**	51	51	51
è (e grave)	54	54	54	54	**D0**	54	54	**D0**	54	54	54
É (E acute)	71	71	71	**E0**	71	71	71	71	71	71	71
ì (i grave)	58	58	58	58	**A1**	58	58	58	58	58	58
ñ (n tilde)	49	49	49	49	49	**6A**	49	49	49	49	49
Ñ (N tilde)	69	69	69	69	69	**7B**	69	69	69	69	69
ø (o slash)	70	70	**6A**	70	70	70	70	70	70	70	70
Ø (O slash)	80	80	**7C**	80	80	80	80	80	80	80	80
ö (o umlaut)	CC	**6A**	CC	**6A**	CC	CC	CC	CC	CC	**A1**	CC
ò (o grave)	CD	CD	CD	CD	**6A**	CD	CD	CD	CD	CD	CD
ü (u umlaut)	DC	**D0**	**A1**	**A1**	DC	DC	DC	DC	DC	DC	DC
ù (u grave)	DD	DD	DD	DD	**79**	DD	DD	**6A**	DD	DD	DD
Ö (O umlaut)	EC	**E0**	EC	**7C**	EC	EC	EC	EC	EC	**5F**	EC
Ü (U umlaut)	FC	**5A**	FC	FC	FC	FC	FC	FC	FC	FC	FC
Ý (Y acute)	AD	AD	AD	AD	AD	AD	AD	AD	AD	AD	**BA**
þ (thorn)	8E	8E	8E	8E	8E	8E	8E	8E	8E	**C0**	8E
Þ (Thorn)	AE	AE	AE	AE	AE	AE	AE	AE	AE	**4A**	AE

Note: The hex values underlined are those which do not match the most common value used for this character.

Appendix G. Windows and OEM codepages

For most single byte languages Windows supports two different codepages. For console output and OEM applications, for example the EDIT application, the OEM codepage (typically 850) is used. For windows applications, for example the NOTEPAD applications, the windows codepage (typically 1252) is used. You can demonstrate the differences by editing the same file using EDIT and NOTEPAD and looking at some special characters (like the pound sterling or an accented character). The characters will display differently in each editor.

If your application uses characters which have different binary values in the two codepages you need to think about which you want to use. If you are using the console to display most of your data then the OEM codepage is the right one, but if you are using mainly windows applications the windows codepage is the right one.

When you create a queue manager the CCSID used for the queue manager is the codepage used by the console. Usually the default for this is an OEM codepage. You can use the command **CHCP** to display the current setting, and can also use it to change the codepage to the windows codepage (for example 1252) if you want. If you want the windows codepage to be used by WebSphere MQ as its default you need to create your queue manager with the queue manger CCSID set to the windows codepage.

When WebSphere MQ sends a message to another system the codepage you send with the message will determine which conversion table is used. If you get characters being converted to unexpected values then this may be because you data has had the wrong CCSID assigned to it. For example, if the data you are sending is created by a windows application but the queue manager has been created with an OEM codepage the default on an MQPUT is to assign the message a OEM CCSID. If your applications are going to create messages in only Windows CCSIDs the best solution is to delete the queue manger, use **CHCP** to change you codepage to the windows codepage, and recreate the queue manager.

If you are using more than one application, and they may be using both OEM and windows codepages, you can override the default CCSID by specifying the message CCSID in the CodedCharSetId filed of the MQMD structure on your MQPUT.

Table 7 shows the OEM and windows codepages for various language groups.

*Table 7. Windows and **OEM codepages***

Description	OEM Codepage	Windows Codepage
USA and Canada	437	1252
Latin 1 used for most Western European languages.	850	1252
Latin 2 used for Eastern European languages.	852	1250
Cyrillic	855, 866	1251
Greek	813	1253
Turkish	857	1254
Hebrew	862	1255
Arabic	864	1256
Baltic rim	921, 922	1257
Note: Not all the OEM codepages are supported by all Windows operating systems.		
Note: For Multibyte CCSIDs used on Windows see "Setting the Queue Manager CCSID on WebSphere MQ for Windows".		

Appendix H. Notices

The following paragraph does not apply to any country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact Laboratory Counsel, MP151, IBM United Kingdom Laboratories, Hursley Park, Winchester, Hampshire, England SO21 2JN. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594, U.S.A.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both.

AIX	AS/400	IBM
MQ	WebSphere MQ	MVS
MVS/ESA	Operating System/400	iSeries
OS/400	z/OS	zSeries
System/390	400	VSE/ESA

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

C-bus is a trademark of Corollary, Inc.

Microsoft, Windows, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation..

Java and HotJava are trademarks of Sun Microsystems, Inc.

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.