Character Data Translation in DataGate/400

(Applies to Release 7.2 and above)

One of the most important and overlooked features of ASNA DataGate is its ability to render character data (database fields and text) in various character encodings, as dictated by platforms and globalization requirements. Using the facilities of the iSeries and Windows, DataGate not only supports transparent iSeries-based EBCDIC character translation, but also the unique international character sets supported by particular platforms. This document discusses iSeries, Windows, and DataGate character translation functionality; past and present. By illuminating DataGate's translation techniques and corresponding facilities, users are better equipped to configure and maintain robust character data applications. This document assumes some familiarity of applications programming with Windows, the iSeries, and DB2/400 database architecture.

International Character Sets

In the traditional green screen environment, developers were not as concerned with character translation issues, because in this hardware-based scheme, data entered into a green screen terminal was only rendered on a green screen terminal (or its report-printing twin). This "panacea" was disrupted by emerging globalization, the Internet, personal computers, and client/server-based access schemes. From very early on however, IBM and other computer vendors had introduced "national language" character sets reflecting local language and culture in international markets. In the case of IBM, these character sets were based on the original eight-bit EBCDIC character code. Later, to penetrate Asian and other markets with larger character sets, IBM introduced the idea of 16-bit character codes, known as Double-Byte Character Sets (DBCS). IBM identifies its various iSeries character encodings with Coded Character Set Identifiers (CCSIDs). Windows likewise supports eight-bit ASCII-based codes, and 16-bit Multibyte Character Sets (MBCS), defined as "code pages." Beginning with Windows 95, Microsoft began abandoning ASCII and MBCS code pages in favor of Unicode. For the most part, Windows and iSeries character data domains are mutually exclusive, and so translation is required when moving data between the two platforms.

In the early 1990's, in an effort to stop the proliferation of proprietary international character sets, an industry consortium proposed a single, unified character set for storing and transferring character data. The Unicode character set defines no less than all known characters from all written languages, and lays down rules for their storage and rendering. Each Unicode character is a 21-bit code point. These codes are mapped into several architecture-friendly encodings. "Unicode" is the name most often associated with the popular UTF-16 encoding, which is a 16-bit encoding. Windows 95 was Microsoft's first effort to bring native operating system support for Unicode encodings. Today, in the .NET framework, character data is manipulated as UTF-16 encoded values. The iSeries introduced some support for Unicode encodings in V3R1. The Java language also uses UTF-16 for character data, so iSeries support has improved with its support for Java. Unicode translations are available via iSeries APIs, but DB2/400 only supports Unicode data in DBCS graphic fields (exposed by DataGate as Unicode fields). The commonly supported Unicode encoding on the iSeries, UCS-2, is nearly identical to UTF-16.

UTF-16 is an extension of UCS-2 which provides support for “surrogate pair” values. The UTF-16 extension was necessary for the rare characters of certain languages such as historical Chinese.

The challenge for middleware systems such as DataGate is to support and exploit each platform’s capabilities to deliver accurate character data to application programs while minimizing the configuration and support effort required. While traditional, eight-bit based character fields of iSeries database files will never be able to store arbitrary international characters, an understanding of the iSeries’ limitations and DataGate’s capabilities allows the creation of useful, globally-aware applications.

Character Data Translation in DataGate

DataGate exposes a database interface familiar to iSeries users. Character data is contained in character and DBCS database fields. DataGate object management interfaces also provide textual metadata in file definitions and object “text” fields. This data is translated on the fly to the native character set of the current execution context. When a DataGate application adds a record to an iSeries database file, the character fields in that record are translated to the character set in use on the server. This character set is defined by the CCSID associated with the iSeries job that DataGate runs under.

In previous versions, ASCII/EBCDIC character translation was driven by eight-bit translation tables derived from the iSeries at connection time. That is, a DataGate client would initiate a connection to the server, announcing its Windows-based code page for character data. The DataGate server would use iSeries APIs and the job CCSID to create simple translation tables that were returned to the client. DataGate client then performed all single-byte character translation using these tables. Effectively, character data sent to and received from the iSeries had an EBCDIC encoding. Figure 1 is a flowchart illustrating this legacy design. Note that older versions of DataGate client software versions accessing 7.2 and newer DataGate servers will continue use this older scheme.

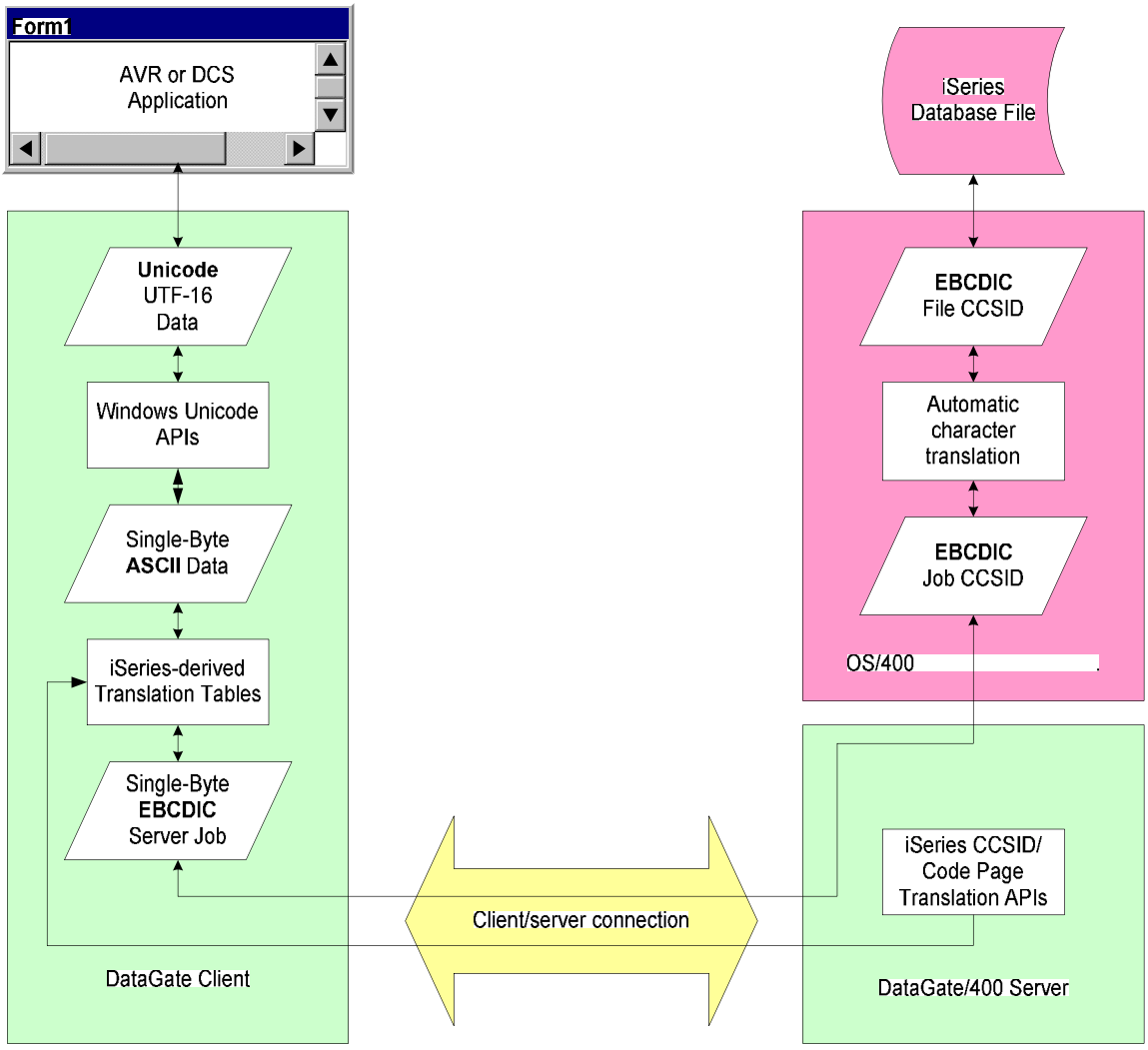

Figure 1. The former DataGate translation scheme (prior to DataGate 7.2) relied on iSeries support for Windows code pages, and did not properly support bidirectional character data.

This scheme has two major drawbacks. First, it relies on iSeries support of various localized Windows system code pages. If the code page for a particular international version of Windows is not supported by the iSeries machine, that Windows platform cannot be used with DataGate/400. Second, DataGate for .NET exposes character fields as Unicode character types, and hence yet another translation layer is required on the client to map EBCDIC-based fields.

DataGate 7.2 introduced a new translation scheme which takes advantage of iSeries Unicode support. This is illustrated in Figure 2. Now, DataGate clients send and receive UTF-16 encoded characters to the DataGate/400 server. DataGate/400 then translates the Unicode data to the EBCDIC encoding of the job. This translation is performed using IBM's iSeries translation APIs, rather than the previously derived tables. One advantage of using iSeries APIs is that "bidirectional" text, as found in Hebrew and Arabic languages, is correctly rendered in the resulting translation.

Also, by standardizing on Unicode, the iSeries platform need only support a translation between its native EBCDIC and Unicode, rather than multiple Windows code pages. On the client, DataGate for .NET natively manipulates the UCS-2 encoded data as UTF-16, with no translation step required.

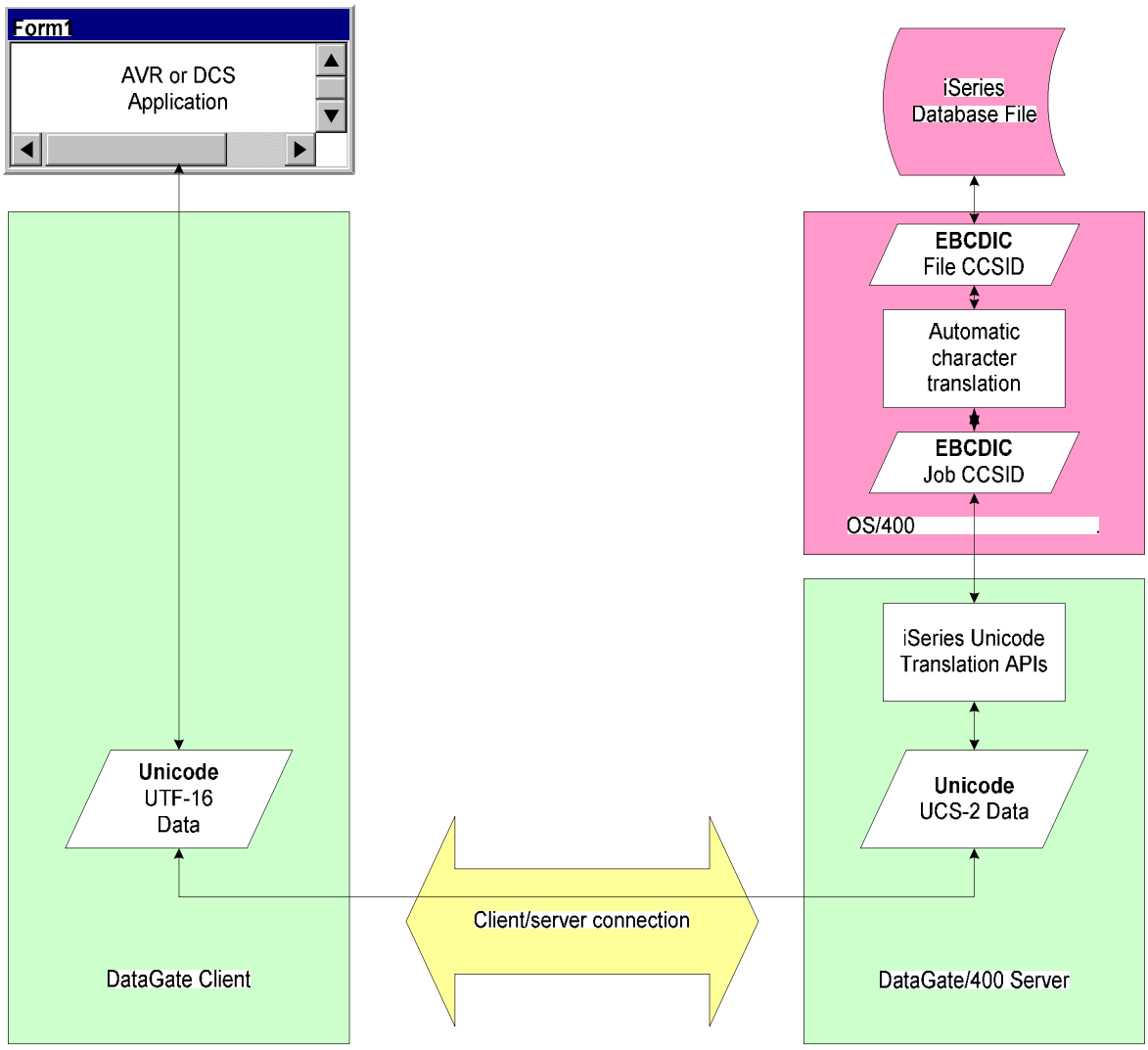

Figure 2. The improved DataGate translation scheme leverages iSeries support for Unicode, and greatly simplifies client-side character data processing.

An important distinction should be made: the DataGate translation scheme uses Unicode character encodings as a client/server transport mechanism, and not as a database storage medium. As mentioned in the previous section, the iSeries does have a notion of "Unicode fields" capable of storing UCS-2 data as DBCS Graphic fields.

While DataGate supports these fields (as the DataGate Unicode field type), it is not dependent on this feature, and is otherwise unrelated to the DataGate character translation scheme.

Multilingual Features in Windows 2000 & XP

Windows includes a Unicode script processor called Uniscribe that supports line measurement, display, caret movement, character selection, justification, and line breaking of Unicode plain text and rich text.

The rules governing the shaping and positioning of glyphs are specified and cataloged within the Unicode standard. The shaping engines that comprise the Windows Unicode Script Processor implement this standard for applications performing complex text layout.
A complex script is one that requires special processing to display and edit because the characters are not laid out in a simple linear progression from left to right, as most European characters are. This special processing falls into several general classes.

- Character Reordering. Characters must be rearranged from logical (keystroke) order to visual order.
- Contextual Shaping. In some languages, the choice of which glyph to display depends on the surrounding characters.
- Display of Combining Characters and Diacritics. Multiple characters must be stacked or combined into one cluster.
- Specialized Word-break and Justification Rules. Some languages require special word-break logic because there is no fixed set of characters that delimit words.
- Cursor Movement and Hit Testing. The mapping between screen position and a character index for, say, selection of text or cursor display requires knowledge of the layout algorithms.

Summary

The implementation of round trip EBCDIC-Unicode translation employed in DataGate 7.2 and above is a rigorous amalgam of the tools provided by IBM and Microsoft, both of whom have constructed their APIs and algorithms under the umbrella of the Unicode Consortium.

Unicode treats characters in two ways: Logical order and Display order. The logical order is the order in which text is typed on a keyboard. Display order is the order of glyphs presented in text rendering. Typically data on the iSeries is stored in Logical order. What is displayed to the end user in Windows or on a browser, however, is dictated by the Unicode text rendering rules, and the two sequences are not always the same.

Even though with this scheme all character data supplied by DataGate/400 is Unicode, not all Windows platforms may be able to render all Unicode characters. For example, Windows XP is capable of using Unicode-encoded Tajik characters internally, but unless Tajik character support is actually installed (or it is a Tajik version of Windows), it won't be able to display the data in a text control.

Appendix I -Translation Scheme System Details

This section is intended to clarify in full systems programming detail the character translation performed by DataGate/400. The procedure uses standard iSeries system APIs, all of which are publicly documented by IBM. These APIs are:

- QtqIconvOpen. Opens a translation session in which to call iconv.
- iconv. Translates a character string from one character set to another.
- iconv_close. Closes a translation session.

Upon initiating a new connection, DataGate/400 calls QtqIconvOpen twice to open two conversion "handles," used in subsequent calls to iconv. The two handles identify 1) a conversion session for translating UCS-2 encoded strings to an EBCDIC character set, and 2) another conversion session to translate job EBCDIC to UCS-2. In both calls to QtqIconvOpen, the current job's CCSID is specified for the EBCDIC character set, and the CCSID 13488 is specified for the UCS-2 character set. The QtqIconvOpen API accepts two structured parameters of type QtqCode_T, which configure the conversion session. DataGate/400 sets the values of these parameters as follows:

QtqCode_T Members	EBCDIC Target/Source	Unicode Target/Source
CCSID	0 (special value indicating current job CCSID)	13488
Conversion alternative	0	0
Substitution alternative	0	0
Shift-state alternative	0	0
Input length option	0	0
Error option	0	0

The two handles remain open until the DataGate/400 job ends, at which time they are closed with iconv_close.

iconv is called to convert all character data crossing the client/server data link. One of the two conversion handles is passed to iconv, depending upon which conversion (EBCDIC to Unicode or Unicode to EBCDIC) is required.

Configuring Applications for iSeries Access

Since the DataGate Client now receives character data from the server in Unicode, configuring the AVR.NET or DCS program environment for a particular character set is generally not required. However, some versions of Windows may not fully support a particular language or culture, regardless of Unicode support. In those cases, a localized Windows release properly supporting a particular language should be considered.

Since Unicode is not the native character set of the iSeries (and thus DataGate/400 must convert Unicode data to and from EBCDIC character sets), some care must be exercised when configuring the iSeries to get reliable results. Basically, an EBCDIC character set must be chosen containing all the characters to be used in your application (or contained in the character fields of your target data). Of course, any EBCDIC character set chosen will be a *subset* of the characters available via Windows, and any characters entered in a Windows application that are not in the EBCDIC set will be saved as unsupported data on the iSeries.

The Datagate/400 server job must be configured to run under the CCSID corresponding to the chosen EBCDIC character set. There is more than one way to influence the CCSID of an iSeries job, e.g., system value QCCSID, but the simplest method is to associate the CCSID with the user profile which the job will run under. For example, if you connect to DataGate/400 as user BILL, you would set the BILL profile's default CCSID (the 'CCSID' parameter of 'CHGUSRPRF').

Configuration Restrictions and Pitfalls

File vs. Job CCSID

Note that the iSeries job CCSID may not define the character set in which the data is finally stored in the database. In both Figure 1 and 2, notice that OS/400 performs an *implicit* character conversion step when reading and writing database file records Figure 3). That is, OS/400 delivers data to (and expects data from) DataGate/400 in the CCSID of the job, even though it may be stored in the CCSID defined by the database file. This can be confusing, especially in systems where more than one CCSID is commonly in use. However, OS/400 will raise an exception in the job if no valid conversion exists between the file and job CCSID, to prevent data corruption.

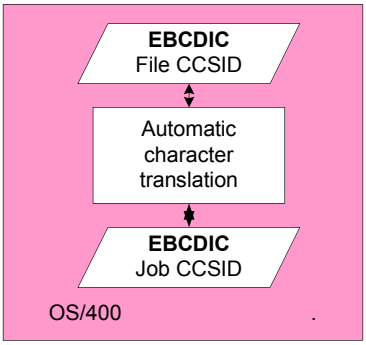

Figure 3. The iSeries automatically converts File CCSID to DataGate job CCSID

UTF-16 vs. UCS-2

Due to the UCS-2 encoding in use on the iSeries, some data loss can occur when using UTF-16 surrogate pair data mapped to standard iSeries character fields. Fortunately, surrogate pair data is currently very rare in common application scenarios (see International Character Sets above). Unfortunately, the iSeries platform will not be able to translate surrogate pairs to useful data in any SBCS EBCDIC character set.

If the target EBCDIC character set is mixed-encoding DBCS, then it may be possible that surrogate pair data in DBCS fields can survive a round trip between the client and server, and thus be usable in AVR.NET. However, other file access (e.g., green screen applications) will not render the data accurately.

For best results when surrogate pair character data is required, it is recommended to limit that data to iSeries/DataGate "Unicode" fields (in DDS file definitions, DBCS Graphic fields defined as CCSID 13488), and that suitable padding be provided in the field length to allow for surrogate pair data.

Bidirectional Data Support

Some languages are intended to be read right-to-left, rather than left-to-right as in English. In fact, in certain cultures, "bidirectional" sentences composed of words with both RTL and LTR orientation are very common. Until Unicode, there was no single, clear definition of the rendering and storage of bidirectional (or bidi) strings. Before Unicode, computer manufacturers and users developed schemes and conventions which were generally not very interoperable. An important feature of the new DataGate translation scheme is that it reliably interoperates with the iSeries bidi scheme much better than the prior scheme.

The iSeries bidi scheme leverages the implicit conversion between the stored CCSID and the job CCSID, providing a so-called "logical view" of character data. An example scenario is bidi data stored in a file as Arabic EBCDIC CCSID 420. With a special Arabic job CCSID (such as 62224), data is automatically converted so that it renders with a certain, typical bidi convention. For detailed information, please consult the iSeries Globalization documentation, with particular attention to "Bidirectional CCSID" topics (see also References below).

DataGate depends upon iSeries Unicode support to correctly translate EBCDIC bidi data to the Unicode standard for bidi, via the iconv system APIs. It is critical that the correct combination of file and job CCSID be used *consistently* to achieve a correct bidi rendering of generically stored data.

DBCS Support

The new DataGate translation scheme improves support for mixed-encoding DBCS fields and text data. In the prior scheme, an entire DataGate/400 installation was required to be dedicated to accessing DBCS data of a particular CCSID. Now a single installation can support both DBCS and SBCS access, with an important restriction: all DBCS character sets in use must be compatible with a single common SBCS character set; that is the SBCS character set must be a subset of the DBCS character sets. The DataGate/400 server user profile, DG8SVCPRF, should be configured to run under the common SBCS CCSID. Also, the password used to authenticate the user must use only SBCS characters.

Appendix II – Compatibility with ASNA COM Products

AVR Classic applications and DataGate Client legacy tools[1] still expect single-byte characters for character fields. Because these products are not Unicode, they may not always render data in the same manner as your AVR .NET application, particularly bidirectional text.

ASNA is unable to further correct the behavior of its COM-based legacy tools. Full Unicode support will be available from the new .NET -based utility products (release date is not available).

There is a registry key to be aware of that drives the translation process for AVR Classic applications and DataGate Client legacy tools as noted below:

Prior to AVR 4.1 (adbcom.dll version < 7)

These versions used the OEMCP registry key value to let DataGate/400 determine which translation tables to fetch in the connection initialization/handshake. These tables are used by adbcom.dll to translate single-byte characters to/from EBCDIC by hand, as required by the older client/server protocol.

OEMCP registry key:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Nls\CodePage\OEMCP

AVR 4.1 (adbcom.dll version = 7)

Unicode characters are converted to single-byte characters of the code page specified by the ACP registry key. The conversion is performed on the client side with the MultiByteToWideChar and WideCharToMultiByte Win32 APIs.

ACP registry key:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Nls\CodePage\ACP

[1] Database file Editor (adbfedit.exe); Database Manager (adbfm.exe); Print File Manager (adbpfm.exe) – All of these products use the COM DataGate Client, adbcom.dll.

References

The following list of references was used in the design of DataGate character translation procedures, as well as to prepare this document. They are given in general order of relevance.

- AS/400 National Language Support V4R2, IBM, 1998, SC41-5101-01, http://publib.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/QB3AWC01/CCONTENTS?DT=19971120215738.
- Unicode Home Page, Unicode Inc., 2005, http://www.unicode.org/.
- "Unicode and Character Sets," Microsoft Corp., 2005, http://msdn.microsoft.com/library/en-us/intl/unicode_6bqr.asp.
- "Globalizing and Localizing Applications," Visual Studio Programmer's Guide for the .NET Framework, Microsoft Corp., 2005, http://msdn.microsoft.com/en-us/library/1021kkz0.aspx.
- OS/400 National Language Support APIs V4R1, IBM, 1997, SC41-5863-00, http://publib.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/QB3AMO00/CCONTENTS?DT=19980807163405.
- "iSeries Bidirectional CCSID Information", iSeries Globalization, IBM, 2002, http://www.ibm.com/servers/eserver/iseries/software/globalization/bidi/main.html
- Unicode Standard Annex #9 The Bidirectional Algorithm, Mark Davis (IBM), 2005
- Microsoft Products & Arabic Support Version 3.0, Microsoft Corp., 2005, http://www.microsoft.com/middleeast/arabicdev/.
- "UTF-16/UCS-2," Wikipedia, 2005, http://en.wikipedia.org/wiki/UTF-16.
- "iSeries CCSID Information," iSeries Globalization, IBM, 2002, http://www.ibm.com/servers/eserver/iseries/software/globalization/ccsid.html.
- AS/400 International Application Development V4R2, IBM, 1998, SC41-5603-01, http://publib.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/QB3AQ501/CCONTENTS.
- Supporting Multilanguage Text Layout and Complex Scripts with Windows NT 5.0, Microsoft, 1998 http://www.microsoft.com/msj/1198/multilang/multilang.aspx.