WebSphere Data Interchange v3.2.1
Source and Target Encoding
for Data Transformation Maps

Overview

As part of WDI 3.2.1 CSD20 and CSD21, the WDI Server has been enhanced to allow users to specify alternate character encodings such as UTF-8 for both source and target data. Alternate character encodings can be specified for Data Formats, EDI, and XML data.

By using alternate encodings, users can correctly read and generate data that contains characters that cannot be represented in the current system codepage. For example, if you are mapping XML data that is in UTF-8 format (a type of Unicode encoding), and it contains Eastern European characters such as **Ą** or Chinese characters such as **?** , then these characters cannot be mapped correctly if your output uses a US-based codepage.

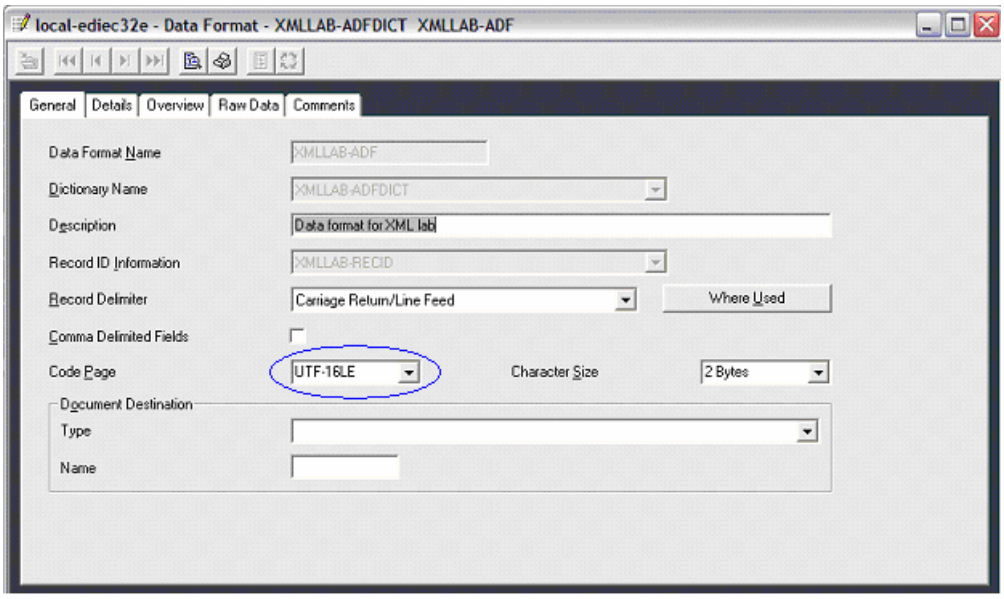
Similarly, when WDI reads the input data, it must know how to interpret the bytes . For example, if the input data is interpreted as "ISO-8859-1", a codepage for Western European languages, then the byte x'AE' would be interpreted as the character ®. However, if the input data is interpreted as "ISO-8859-2", an Eastern European codepage, then it would be interpreted as the character **Ž**.

This document describes how WDI determines the source document encoding for Data Formats, EDI, and XML data, and how users can specify the encoding to be used for target Data Format, EDI, and XML documents.

Data Format (Record-oriented or flat file) Data

Determining the encoding of the source Data Format document

The encoding of the source Data Format document is determined from Code Page value in the Data Format definition.



The method to specify the encoding is unchanged from previous CSD releases. However, many of the limitations in previous CSDs have been resolved in CSD21. Before, certain functions were not supported if the input data was not in an encoding that was "similar" to the local codepage. For example, when the source data was in UTF-16 format WDI was not able to:

- Split multiple transactions in an input file based on the header or trailer record. Instead, the entire input file was treated as a single transaction.
- Extract trading partner information such as the sender and receiver id/qualifier.
- Process characters that were outside the local codepage (for example, Japanese or certain Eastern European characters), due to some internal conversions that occurred.

These limitations, plus some other codepage-related issues are fixed in CSD21. UTF-16 and other encodings are now supported for Data Format (record-oriented) data.

Setting the encoding of the target Data Format document

The encoding for the target Data Format document can be specified either by setting the **EncodeTarget** property in the map, or by setting the Code Page value in the Data Format definition. If the **EncodeTarget** property is set, it will override the Code Page value. If neither the **EncodeTarget** property nor the Code Page value is specified, the output Data Format document is created using the default system codepage.

Field lengths and record id offsets

Prior to CSD21/FP17, the field length and record id offsets always had to be specified in terms of bytes. For example, if you specified a Code Page value of "UTF-16" which uses 2 bytes for each character, and you had a 10-character field, you would have needed to specify a field length of 20 (10 characters x 2 bytes per character). Similarly, record id offsets needed to be adjusted if the character length was more than one byte. In CSD21/FP17, a new field was added to the Data Format definition, and another one to the Record ID Information to make this easier.

In CSD21/FP17, a Character Size field was added to the Data Format definition. This field defines the number of bytes per character for the data format, and is used to help calculate the byte offset and length of the fields. The value can be either "1 Byte" or "2 Bytes". The default is "1 Byte".

For character-based fields, the field length is multiplied by the character size to determine the number of bytes in the field. Character-based data types include:

- A = Alphabetic
- AC = Application Control
- AN = Alphanumeric
- CH = Character
- DT = Date
- FN = File Name
- HX = Hexadecimal
- ID = Identifier
- N = Numeric
- Nn = Numeric
- PW = Password
- R = Real
- Rn = Real
- TM = Time

Binary data types are not adjusted for the character size. Binary data types include:

- BN = Binary
- Bn = Binary
- In = Integer
- Hn = Hexadecimal
- IT = Integer
- IV = Incrementing Value
- Ln = Zoned Decimal
- Pn = Pack Decimal
- PD = Packed Decimal
- Zn = Zoned Decimal
- ZD = Zoned Decimal

For single-byte codepages, the Character Size value should be 1. For 16-bit encodings such as UTF-16, this value should be 2. For mixed-length encodings such as UTF-8, this should be set to 1, and the data should be exactly the specified number of bytes for fixed-length data formats.

In addition to specifying the Character Size for the Data Format definition, you can also specify the Position Type for the Record ID Information. This indicates whether the record ID position is in terms of characters, bytes, or fields. The following position types are valid:

- Character – The offset indicates the number of *characters* from the beginning of the record (1 indicates the record id starts with the first character). The position are multiplied by the character size for the data format to determine the byte offset. If the record id data type is character-based, then the record id length is multiplied by the character size to determine the record id length in bytes.
- Byte - Number of *bytes* from beginning of the record (1 indicates the record id starts with the first byte). No character size adjustment is made to the position or length.
- Field - This is only valid if the record id is used for a comma delimited data format. The position indicates a *field number* from the beginning of the record. For example, if position = 3, this would indicate that the record id is the 3rd field in the comma delimited record, regardless of the byte or character offset. The record ID length is adjusted by the character size to determine the length in bytes.

The default position type is *Character*. If the position type is set to *Field* and the Data Format is not comma-separated, then position type *Character* will be used instead.

XML Data

Determining the encoding of the source XML document

The encoding of the source XML document is determined from the byte-order mark and XML declaration, according to the rules in the W3C XML Recommendation. This is unchanged from previous CSD releases .

For example, the following would indicate that the data is encoded using codepage ISO 8859 -2:

```
<?xml version="1.0" encoding=”iso8859-2” ?>
```

For more details, see the W3C XML Recommendation (http://www.w3.org/TR/2004/REC-xml-20040204/), Appendix F. Autodection of Character Encodings.

For z/OS platforms, users can override the autodetection logic using the XMLEBCDIC keyword on the PERFORM command. This forces WDI to interpret the XML input using the IBM EBCDIC codepage 1047, regardless of the encoding value specified in the XML declaration . This is useful when the XML data has been converted to EBCDIC from its original encoding when it was transferred to the z/OS system using FTP, WebSphere MQ, or some other mechanism. Again, this is unchanged from previous CSD releases, and is described in more detail in the WDI Programmer’s Reference, in the “XML encoding considerations for z/OS” section.

Setting the encoding of the target XML document

The encoding for the target XML document can be specified by setting the **EncodeTarget** property in the map. If the **EncodeTarget** property is not set, the output XML is created using the default system codepage.

For example, if the XML output is always going to be written in UTF-8 format, the following command can be added to the map:

```
SetProperty ("EncodeTarget", "UTF-8")
```

If different encodings are to be used for different trading partners, the EncodeTarget value can be set based on the trading partner properties. See "Using trading partner properties to determine the target encoding" below.

If you use an encoding that cannot be determined using the normal autodetection logic, such as "ISO-8859-2", you should also set the "DIProlog" property to specify an appropriate XML declaration. Otherwise, the output XML data may not be interpreted correctly by other users or applications.

EDI Data

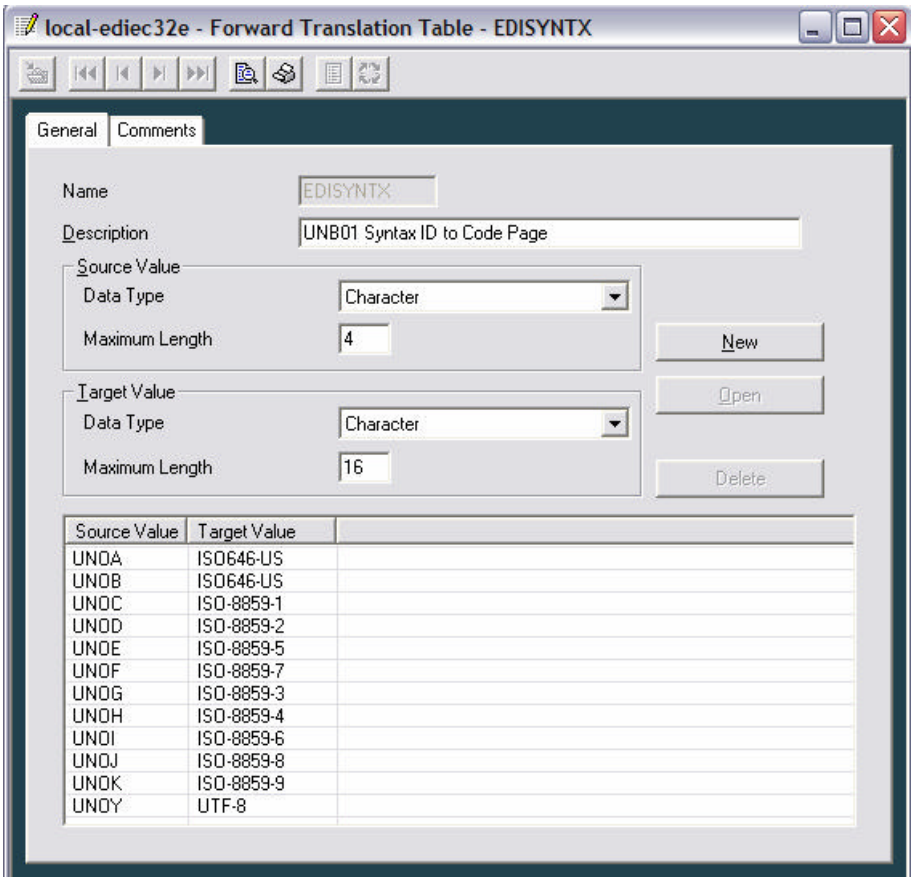
Determining the encoding of the source EDI document

X12, UCS, and UN/TDI source documents are always interpreted based on the system default codepage. These EDI standards have relatively limited character sets, and do not currently specify a mechanism to use alternate character encodings.

The EDIFACT standard does allow users to specify a different encoding for the data by using the Syntax ID element in the UNB segment. To determine the encoding of a source EDIFACT document, WDI will:

- Read the Syntax Id (UNB0101) of the source document.
- Look up the value in the EDISYNTX translation table
- If a matching "Source Value" is found in the table, then the "Target Value" is used to interpret the data.
- If no matching "Source Value" is found, then the system default codepage is used to interpret the data.

For example, the following EDISYNTX table is provided as an EIF file in the "samples" directory for the Windows and AIX builds. This may be imported and used as -is, or it may be customized.



If the source EDIFACT document had a Syntax id value “UNOD”, then WDI would look up “UNOD” in the table above and find that it corresponds to encoding “ISO-8859-2”. Then encoding “ISO-8859-2” would be used to read the source document. If “UNOD” were not found in the table, WDI would read the source document using the default system codepage.

If a user wants to read all EDIFACT documents using the same encoding, they can set all valid (or expected) target values to the desired encoding. For example, if all EDIFACT documents are to be read as UTF-8, then the user can just set all “Target Values” to “UTF-8” in the table above. Although this is contrary to the EDIFACT rules for determining the encoding, it may be appropriate if the user has an agreement with their trading partner, or if another application is converting the data from the original encoding to a common encoding.

Setting the encoding of the target EDI document

The encoding for the target EDI document can be specified by setting the **EncodeTarget** property in the map. If the **EncodeTarget** property is not set, the output EDI is created using the default system codepage.

For example, if the EDI output is always going to be written in UTF-8 format, the following command can be added to the map:

```
SetProperty ("EncodeTarget", "UTF-8")
```

If different encodings are to be used for different trading partners, the EncodeTarget value can be set based on the trading partner properties. See "Using trading partner properties to determine the target encoding" below.

You should generally set the Syntax ID in the output data so it corresponds to the encoding that is used. For example, if you set the EncodeTarget property to "ISO-8859-2", you would typically set the Syntax ID to "UNOD" to be consistent with the EDIFACT standard.

Encodings supported

Internally, WDI uses "International Components for Unicode" (ICU) to convert from one encoding to another for Windows and AIX, and uses the C library function iconv() do the conversions on z/OS. As a result, many different encodings are available, although only a subset have been tested and are supported.

The encoding names used are based on the names registered with IANA (Internet Assigned Numbers Authority), although not all registered names are supported by the underlying ICU and iconv() library functions. The full set of IANA-registered names is at:

http://www.iana.org/assignments/character-sets

A list of the encodings supported by the International Components for Unicode is at:

http://www-950.ibm.com/software/globalization/icu/demo/converters

The following encoding names have been tested and are supported for Data Formats, EDI, and XML data on Windows and AIX.

- ISO-8859-1
- ISO-8859-2
- ISO-8859-3
- ISO-8859-4
- ISO-8859-5
- ISO-8859-6
- ISO-8859-7
- ISO-8859-8
- ISO-8859-9
- UTF-8

On z/OS, ASCII-based codepages such as the ones above are not supported for EDI data. UTF-8 may be used for Data Formats and XML data, but has certain limitations (see below). Other EBCDIC-based codepages such as "IBM-037" or "IBM-875" are supported for Data Formats, XML and EDI data on z/OS.

In addition to the encodings listed above, 16-bit encodings such as "UTF-16", "UTF-16LE", "UTF-16BE", and "UCS-2" are supported for Data Formats and XML (all platforms), but are not supported for EDI.

Special restrictions for Data Formats:

- UTF-8 and similar mixed length encodings are supported for Data Formats, but for fixed-length records the data must still appear at consistent byte offsets. For example, suppose you define a fixed length record with a 10 character field. If the Code Page is UTF-8, different characters are encoded using a different number of bytes. Characters such as A - Z only take one byte, while many non-English characters take 2 or 3 bytes. This means that the 10 characters may take anywhere from 10 to 30 bytes. Since this is a fixed-format record, the byte length for the field must always be the same. In this case, you would need to define the Character Size as 1, allow the maximum number of bytes for the field length (30), and pad with spaces if the full 30 bytes was not needed. Because mixed-length encodings such as UTF-8 can be difficult to use with fixed-format records, a fixed-length encoding such as UTF-16 is generally a better way to handle international character sets with fixed-format records.

Special restrictions for XML data:

- If input XML data uses a 16-bit encoding or an encoding that is incompatible with the system default codepage (i.e., uses an EBCDIC-based codepage on Windows or AIX, or an ASCII-based codepage on z/OS), WDI will treat the entire input file as a single XML document. WDI will **not** split the input file into multiple XML documents based on the XML declaration, and will not split a single XML document into multiple documents based on the "Document split" elements.
- XML tag names must be able to be represented in the system default codepage. For example, if your system default codepage is Windows codepage 1252, you should not use Chinese characters in the XML element names. However, characters outside the system default codepage can be passed through in the XML values just fine, as long as both your source and target encodings support them.

Special restrictions for EDI data:

- Input and output EDI data cannot use 16-bit encodings or encodings that are incompatible with the system default codepage (i.e., an EBCDIC-based codepage on Windows or AIX, or an ASCII-based codepage on z/OS). Use of these encodings for EDI data may prevent WDI from recognizing or generating certain types of data correctly, such as numeric values and some segment tags.
- When transaction store images are displayed in the WDI Client, they are displayed in the Windows codepage – not necessarily the encoding used to translate the data. (Note: This only applies to the user interface. Transaction images read from the transaction store for

translation or enveloping will be processed using the encoding rules described previously.)

- When using the transaction store for deferred enveloping, non-English characters (outside the US-ASCII range on Windows/AIX) are not supported for data in the envelope segments (i.e., UNB, UNG, UNH), particularly if alternate encodings are used.
- UTF-8 will not be supported for X12 or UCS (these EDI standards restrict the character sets to Latin characters, so Unicode encodings are not needed.)
- For UTF-8 data, delimiters must be single-byte characters. For example, in UTF-8 the character § is represented by the bytes xC2 xA7, so this character should not be used as a delimiter. (Note: If you are using encoding ISO-8859-1, the same character is represented by the single byte xA7, so it would be an acceptable delimiter in that case.)
- Because the database, including the CHARSET and ALPHANUM tables are defined in "local codepage", there are some limitations on validation of EDI data containing characters outside this character set.
    - If the data contains characters that cannot be represented by the system default codepage, validation level 2 should not be done.
    - Codelist lookups can be done, as long as the value being looked up does not contain characters that are not in the system default codepage.
    - Other types of validation (mandatory segments and elements, min/max length checks, etc.) will still be done, regardless of the characters or encoding of the data.
- For EDI data, the encoding name is restricted to 16 characters. Since most encoding types have multiple aliases defined, you may need to choose one of the shorter alias names.

Other Notes and restrictions

- The WDI 3.2.1 database is in the system default codepage. Therefore, any data that is looked up in the database must be able to be represented in the local codepage. This includes things like trading partner ids/qualifiers, codelists (used for the "Validate" mapping function) and translate tables (used for the "Translate" mapping function). Other syntax-specific examples are included in the Data Format, XML and EDI sections above. Changing your database codepage to use a multi-byte encoding such as UTF-8 is not supported, as this can introduce other problems.

Using trading partner properties to determine the target encoding

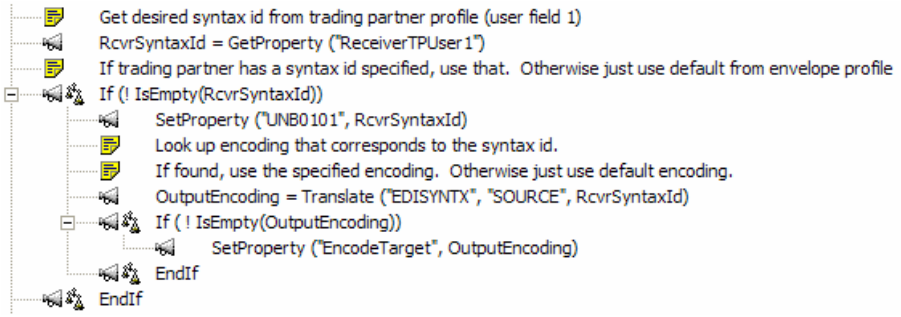
In some cases, users may want to set different target encodings based on the trading partner the document is going to. For example, a user may want to set the "EncodeTarget" property to ISO-8859-2 for their Eastern European trading partners, but ISO-8859-1 for their Western European customers.

To allow users to use trading partner information in their maps, the following new mapping properties from the sender and receiver trading partner profiles are now available. These can be

retrieved using the GetProperty mapping function in a map, and can be used for any purpose – not just to set the target encoding:

Property name	Property Description
SenderTPProfile	Sender trading partner profile name. If no profile is found for the sending trading partner, the value 'UNKNOWN' is returned.
ReceiverTPProfile	Receiver trading partner profile name. If no profile is found for the sending trading partner, the value 'UNKNOWN' is returned.
SenderTPUser1-SenderTPUser10	User fields from the sender's TP profile. If no sender TP profile was found, the user fields from the 'UNKNOWN' trading partner profile will be used if it exists. If the 'UNKNOWN' profile does not exist, the user fields from the 'ANY' trading partner will be used.
ReceiverTPUser1-ReceiverTPUser10	User fields from the receiver's TP profile. If no receiver TP profile was found, the user fields from the 'UNKNOWN' trading partner profile will be used if it exists. If the 'UNKNOWN' profile does not exist, the user fields from the 'ANY' trading partner will be used.

As an example, if you are mapping from XML -> EDIFACT, and want to set the Syntax ID (UNB0101) and target encoding based on the trading partner, you can set User Field 1 to the Syntax ID, then add mapping commands similar to the following to your map:



These mapping commands will:

- Get the Syntax ID from the trading partner profile (user field 1)
- If the Syntax ID is specified:
  - Set the "UNB0101" property, which WDI will use to create the EDIFACT UNB segment.
  - Look up the Syntax ID in the EDISYNTX translate table.
  - If the Syntax ID is found, it will set the target encoding to the corresponding encoding. If not found, it will just let WDI use the default encoding.
- If no Syntax ID is specified in the trading partner profile, it will let WDI use the default Syntax ID (from the envelope profile) and encoding.

Note: The WDI Client allows you to customize the trading partner profile user fields by setting your own display names. For example, you can define the display name for "User field 1" to be "Encoding". To customize the trading partner user fields, select the "View" action, then "Customize". If you customize the display names, the mapping property does not change. However, using the customized display names makes it easier to remember which field is used for which purpose when you edit the trading partner profile. When the list of available property values is displayed for the GetProperty function, the customized name will be shown in brackets next to the property name.

Setting the CCSID in outbound MQ Headers

When WDI sends data out over WebSphere MQ, it generates an MQMD header for the message, and optionally an RFH2 header. Each of these headers includes a Coded Character Set ID (CCSID), which is used to describe the character encoding for the data.

Prior to CSD21, WDI always set the CCSID to the default value to indicate that the character encoding matched the CCSID for the queue manager. Now that WDI allows users to encode the data differently for different output messages, this default CCSID may not be correct.

In CSD21, WDI sets the CCSID in the MQMD or RFH2 header based on the encoding used to generate the output data (i.e., the EncodeTarget property or the Code Page from the Data Format definition). If only an MQMD header is included in the MQ message, then the CCSID in the MQMD is set to indicate the character encoding for the translated data. If an RFH2 header is included, then the CCSID in the MQMD is set to the default value (it describes the RFH2 header), and the CCSID in the RFH2 header is set to indicate the character encoding for the translated data.

Because the encoding specified by the EncodeTarget or the Data Format definition is a string, and the CCSIDs in the MQ headers are integer values, a conversion needs to take place. The following steps are used to determine the CCSID from the encoding name:

1. The encoding name will be looked up in the ENC2CCS translate table. If found, the translated value is converted to an integer and used for the CCSID. For example, if the EncodeTarget property was set to "UTF-8", and the translate table entry has source value="UTF-8" and target value="1208", then the CodedCharsetId value will be set to 1208.
2. If the CCSID is not found above, and the encoding name is in the format "ibm-nnnn", the CCSID will be set to the "nnnn" value. For example, if the EncodeTarget property was set to "ibm-1208", the CCSID will be set to 1208.
3. If the EncodeTarget property is not set, or if the value is not found in the steps above, then the CodedCharsetId will be set to MQCCSI_Q_MGR (in the MQMD) or MQCCSI_INHERIT (in the RFH2 header, if used) as it is today.

MQ Advanced Adapter -
Controlling codepage conversion when receiving data

By default the MQ Advanced Adapter receives messages from MQ using the “get-with-convert” option. This converts the data to the CCSID defined for the queue manager. If the data is in a different encoding and contains characters that cannot be represented in the default codepage, then these characters may be lost. For example, if the data comes in as UTF-8 and contains Japanese characters, but the queue manager CCSID is for US-ASCII, then the Japanese characters would be lost.

In CSD21, WDIService, part of the WDI Advanced Adapter, allows external control over data conversion through the "wdi.properties" file. Two new properties now appear in the file: "CCSID" and "Convert".

- The “Convert” option determines if WMQ conversion tables should be used, and it is set to either "yes" or "no." The default is "Convert=yes."
- The "CCSID" option overrides the typical conversion to the Queue Manager's Local Code Page. It has a numeric argument that corresponds to the Coded Character Sets as defined by IBM.

In addition to their use in the wdi.properties file, these options may be specified on the WMQ Queue definition in the Trigger Data property or in the WMQ Process Definition as User Data. When specified as part of the WMQ objects, the options supply values in parenthesis; for example, it uses CCSID(1208).