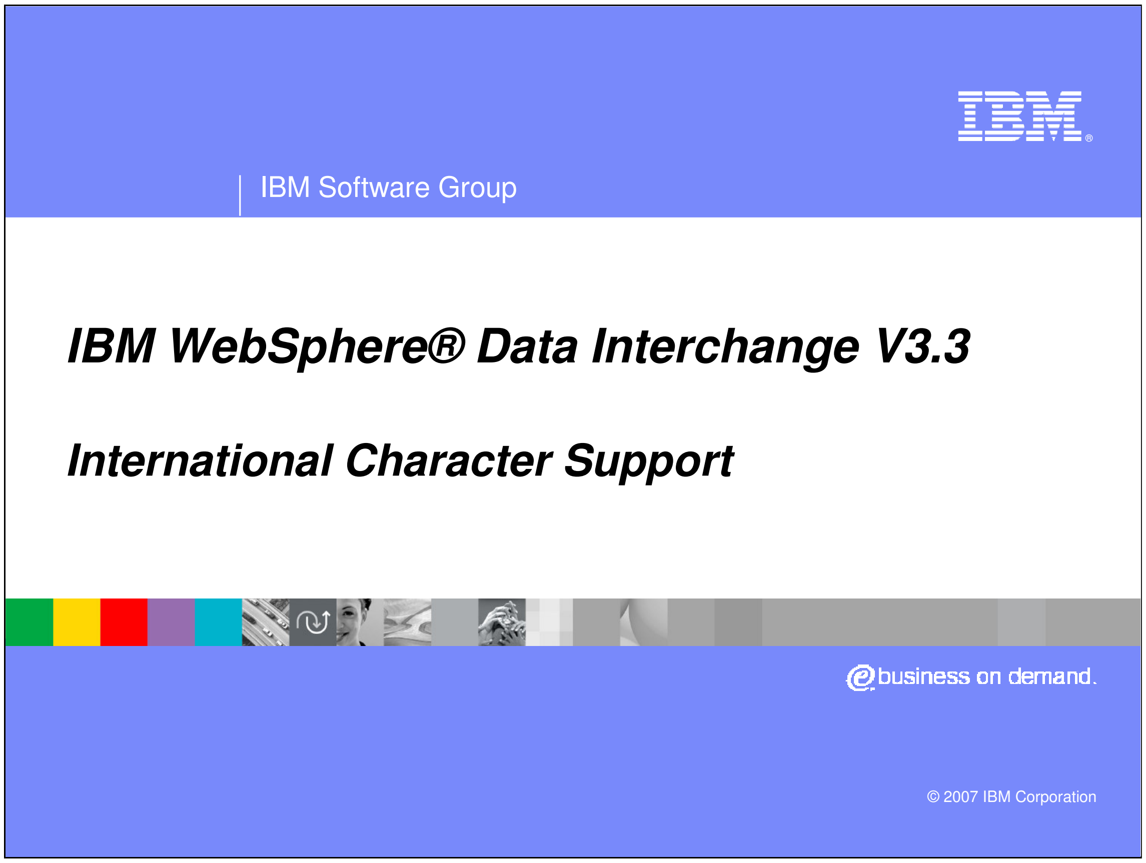
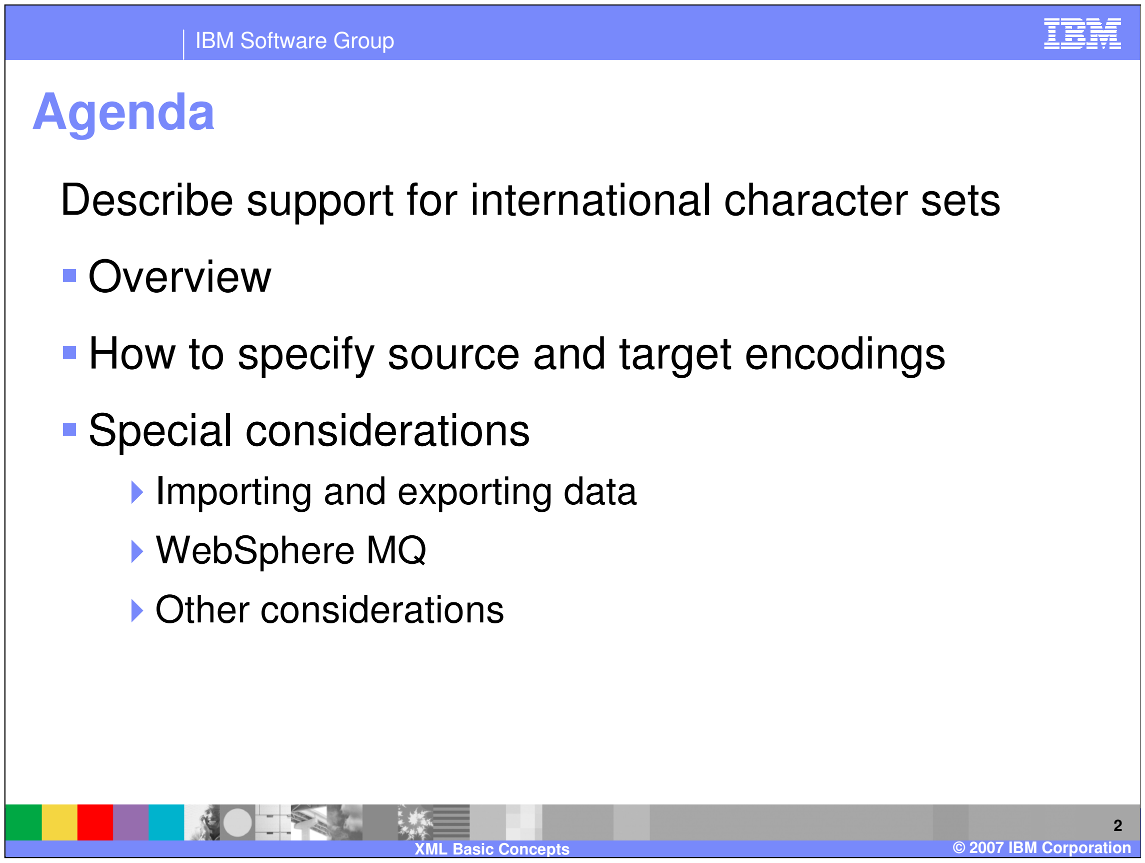


IBM Software Group

IBM WebSphere® Data Interchange V3.3

International Character Support

business on demand.

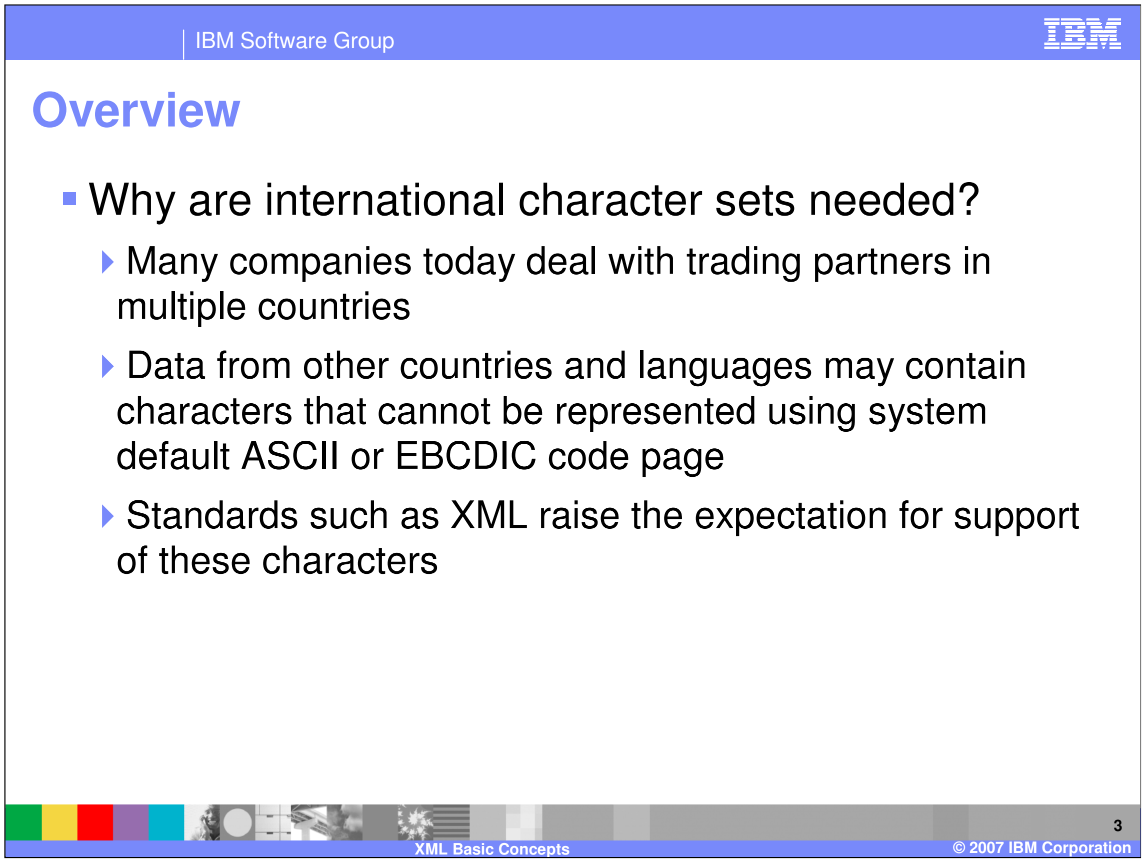


Agenda

Describe support for international character sets

- Overview
- How to specify source and target encodings
- Special considerations
  - Importing and exporting data
  - WebSphere MQ
  - Other considerations

This presentation describes the WebSphere Data Interchange support for international character sets. It starts with an overview of why international characters may be needed. Then it describes how you can specify the encoding for your source and target data. Finally, it will mention some special considerations, including import/export, use with WebSphere MQ, and other considerations.


IBM Software Group

Overview

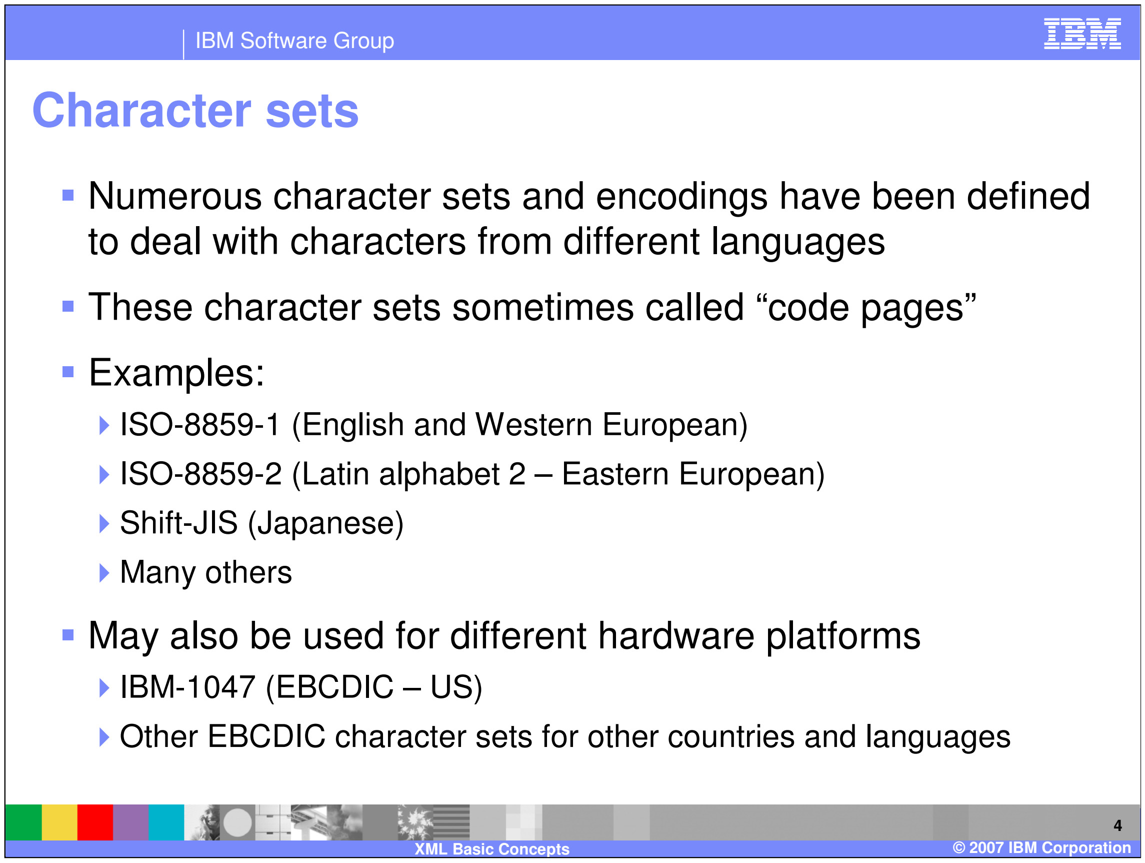
- Why are international character sets needed?
  - Many companies today deal with trading partners in multiple countries
  - Data from other countries and languages may contain characters that cannot be represented using system default ASCII or EBCDIC code page
  - Standards such as XML raise the expectation for support of these characters

3

XML Basic Concepts © 2007 IBM Corporation

In today’s world, companies frequently deal with trading partners from multiple countries - and not just ones where English is the predominate language. This means you may receive data from your trading partners which contains characters that are not included in the default ASCII or EBCDIC code page for your system. Even if much of the data is in English, there may still be special characters in values such as names, addresses, and descriptions. This presentation will refer to these characters as “international characters”.

Standards such as XML offer built-in support for international character sets as a core part of the specification. The EDIFACT standard also allows users to specify character sets besides the default ASCII or EBCDIC code pages. Since these capabilities are included in the standards, it increases the expectation that the data – including characters from any language - can be processed without losing the data. Even if your system cannot display these characters correctly on the screen, you may still want to pass the data intact to other applications that can process them.



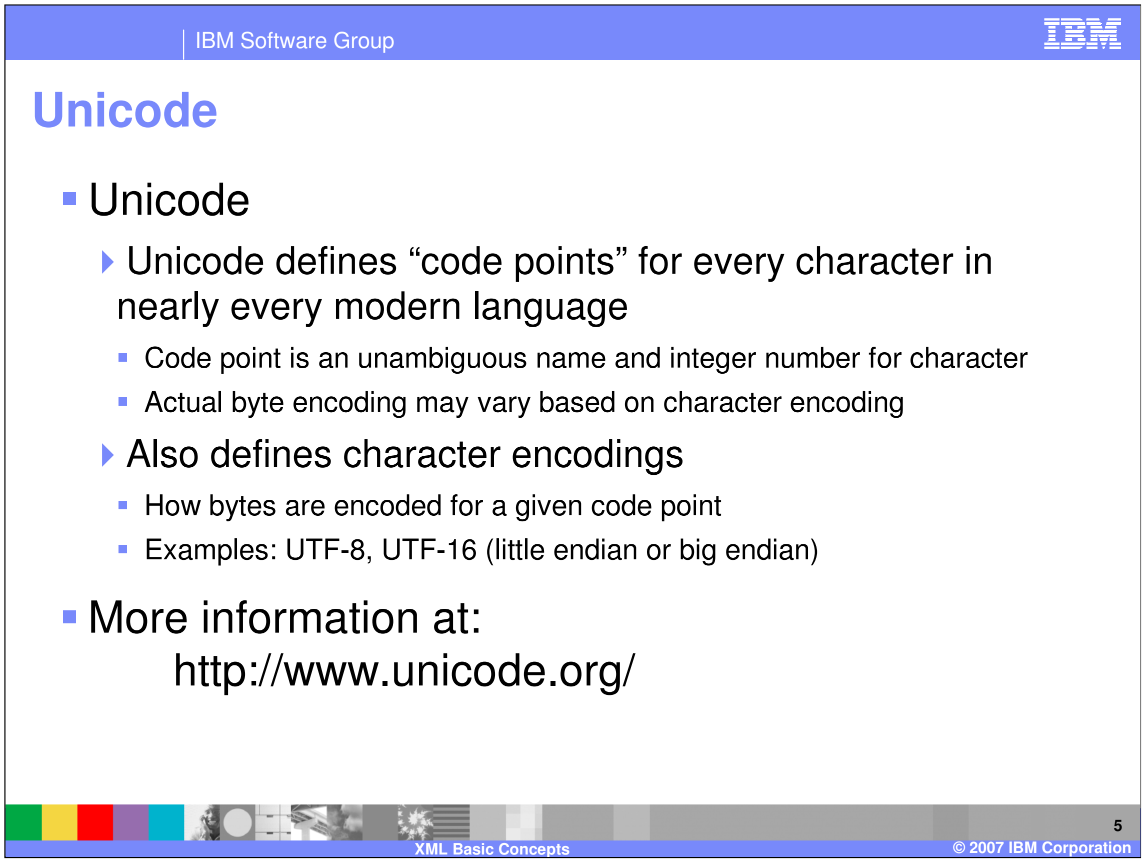
Character sets

- Numerous character sets and encodings have been defined to deal with characters from different languages
- These character sets sometimes called "code pages"
- Examples:
  - ISO-8859-1 (English and Western European)
  - ISO-8859-2 (Latin alphabet 2 – Eastern European)
  - Shift-JIS (Japanese)
  - Many others
- May also be used for different hardware platforms
  - IBM-1047 (EBCDIC – US)
  - Other EBCDIC character sets for other countries and languages

Although a complete discussion of character sets and encodings is far beyond the scope of this presentation, a short discussion of character sets and Unicode may be helpful.

Many different character sets and encodings have been defined over the years to deal with characters from different languages. These character sets are sometimes called "code pages". One example is ISO-8859-1, which includes most of the characters used by English and other Western European languages. Another is ISO-8859-2, which covers more of the Eastern European characters. Some languages like Japanese require more than 256 characters, so special encodings such as Shift-JIS were created to handle these languages.

The use of a particular character set is sometimes based on the hardware platform, rather than the language. For example IBM-1047 is an EBCDIC codepage, and is typically used on IBM mainframe computers. Other EBCDIC character sets may be used on mainframe computers for other countries and languages.


IBM Software Group

Unicode

- Unicode
  - Unicode defines “code points” for every character in nearly every modern language
    - Code point is an unambiguous name and integer number for character
    - Actual byte encoding may vary based on character encoding
  - Also defines character encodings
    - How bytes are encoded for a given code point
    - Examples: UTF-8, UTF-16 (little endian or big endian)
- More information at:
  http://www.unicode.org/

5

XML Basic Concepts

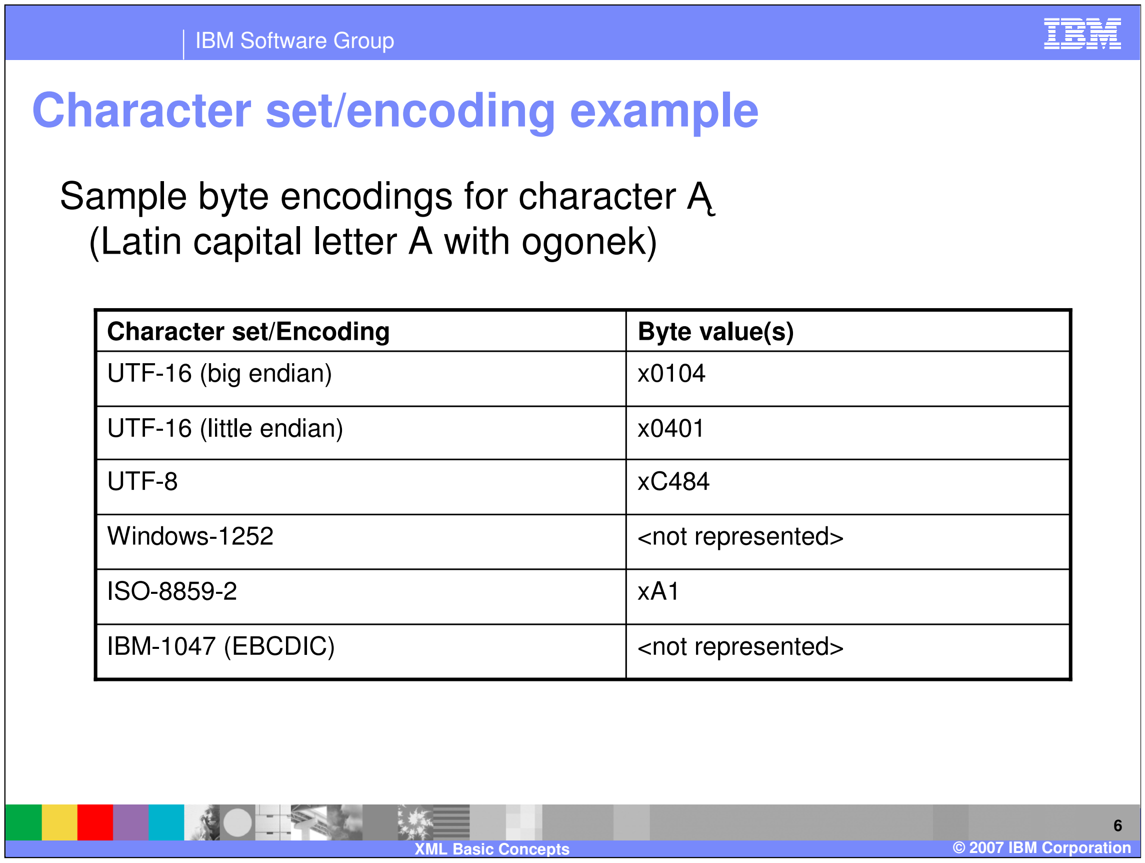
© 2007 IBM Corporation

The Unicode standard is an attempt to reduce the confusion caused by all of the different character sets and encodings.

Unicode defines “code points” for every character in nearly every modern language – and also numerous special characters. The code point is an unambiguous name and an integer number for the character.

These code points can be algorithmically converted to a byte encoding, depending on the character encoding that is used. Examples of different character encodings include UTF-8 and UTF-16. These different character encodings offer different advantages and disadvantages, such as smaller total data size or fixed-length characters. UTF-16 may be encoded as either “big-endian” or “little-endian”, which describes the order that the bytes appear. Little-endian may be preferred for certain platforms such as Windows, while big-endian may be preferred for others.

More information is available at the Unicode web site.

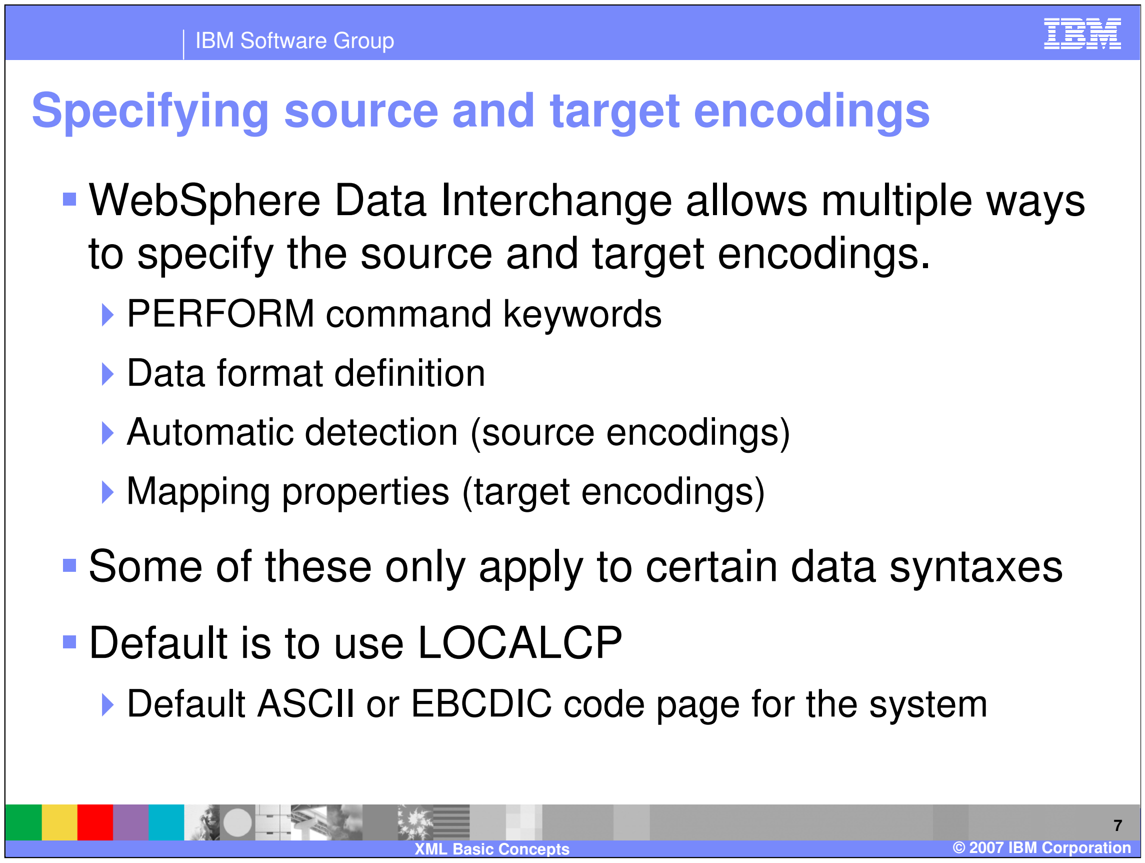


Character set/encoding example

Sample byte encodings for character Ą
(Latin capital letter A with ogonek)

Character set/Encoding	Byte value(s)
UTF-16 (big endian)	x0104
UTF-16 (little endian)	x0401
UTF-8	xC484
Windows-1252	<not represented>
ISO-8859-2	xA1
IBM-1047 (EBCDIC)	<not represented>

This example shows how a particular character may be represented by a variety of different byte values depending on which character set and encoding is used. In some character sets, it may not even be possible to represent a given character. For the UTF-8 and UTF-16 Unicode encodings, the code point assigned to the character is the same, x0104, but the byte encoding is different from one encoding to the next.



Specifying source and target encodings

- WebSphere Data Interchange allows multiple ways to specify the source and target encodings.
  - PERFORM command keywords
  - Data format definition
  - Automatic detection (source encodings)
  - Mapping properties (target encodings)
- Some of these only apply to certain data syntaxes
- Default is to use LOCALCP
  - Default ASCII or EBCDIC code page for the system

WebSphere Data Interchange allows multiple ways to specify the source encoding for your input data, and the target encoding for your output data. The source encoding tells how the input data should be interpreted. The target encoding tells how the output data should be written.

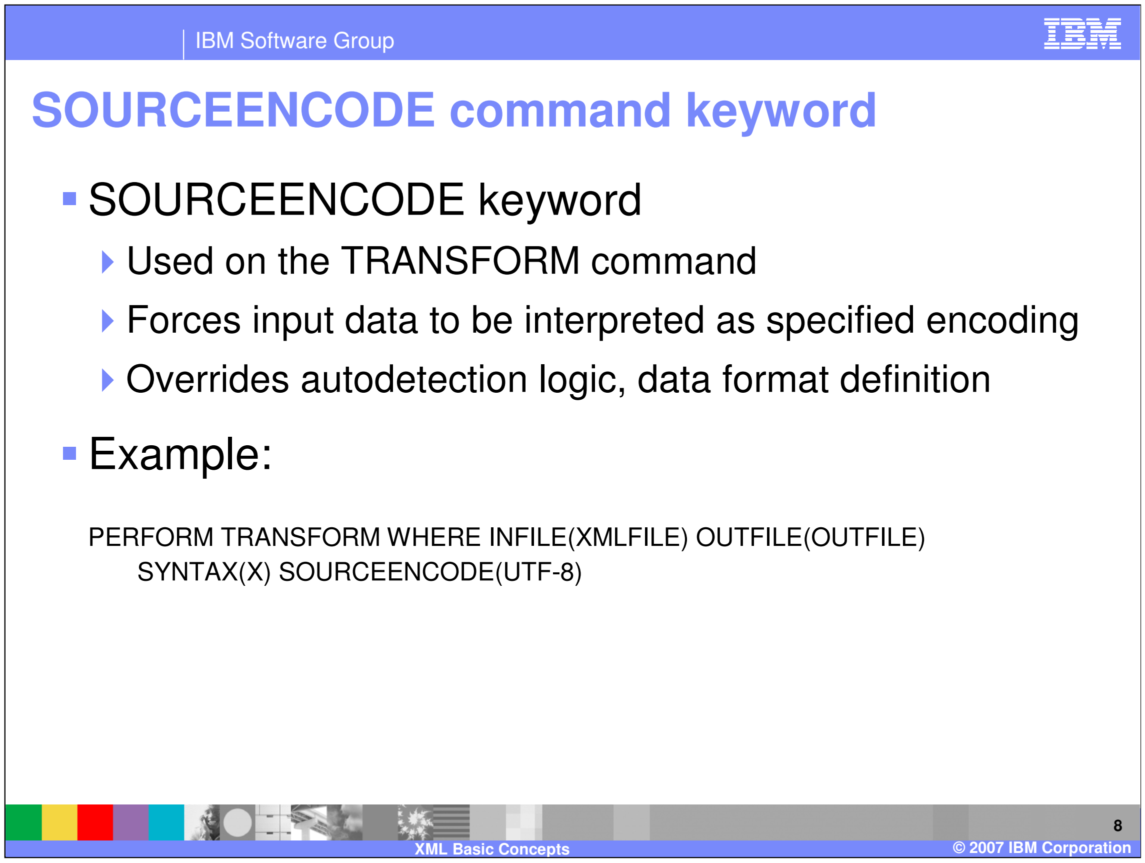
•You can specify keywords on the PERFORM command

•You can define the encoding on a data format definition. This can only be used for data format input and output.

•For certain types of input data, you can let WDI determine the source encoding as it parses the data

•For output data, you can specify the target encoding as a map property

If no encoding information is provided, the default ASCII or EBCDIC codepage is generally used.



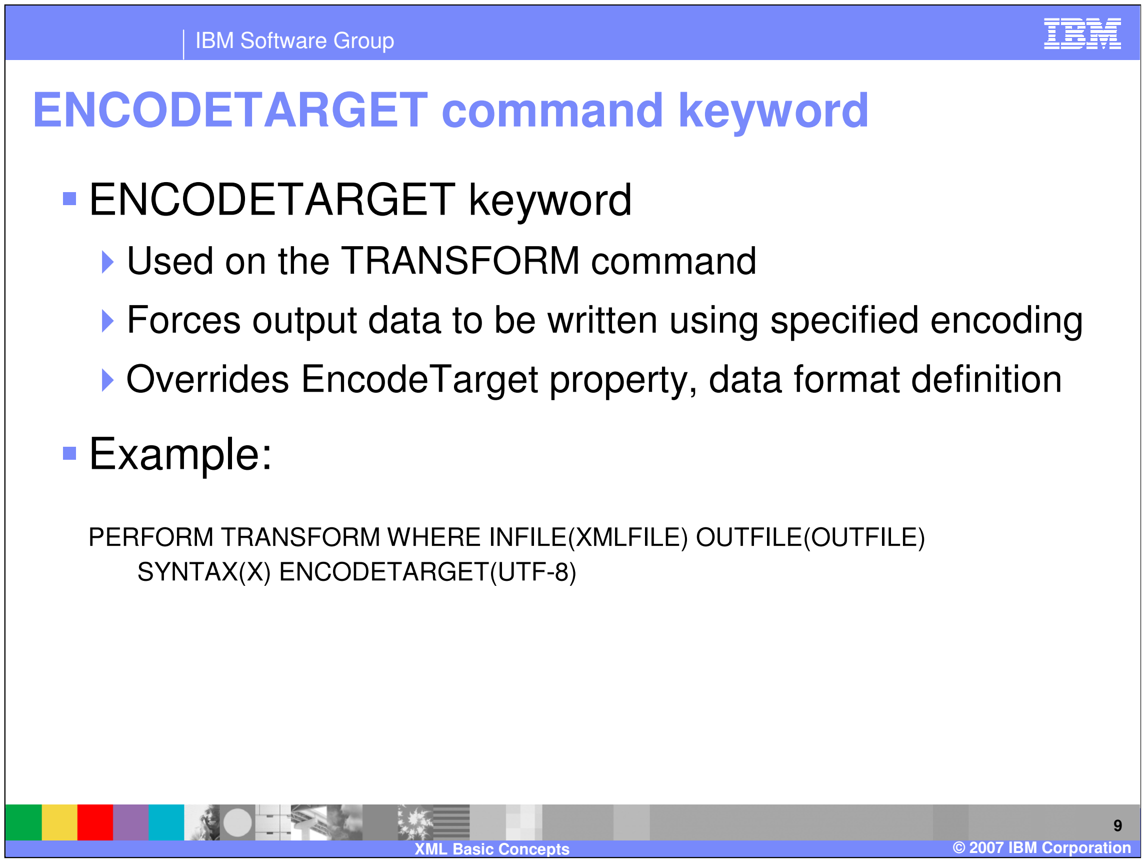
SOURCEENCODE command keyword

- SOURCEENCODE keyword
  - Used on the TRANSFORM command
  - Forces input data to be interpreted as specified encoding
  - Overrides autodetection logic, data format definition
- Example:

```
PERFORM TRANSFORM WHERE INFILE(XMLFILE) OUTFILE(OUTFILE)
     SYNTAX(X) SOURCEENCODE(UTF-8)
```

The SOURCEENCODE keyword can be specified on the TRANSFORM command to force the input to data to be interpreted using a particular encoding. This will override the normal autodetection logic or the code page specified in the data format definition. This can be useful if you know that an application will always pass the data to WebSphere Data Interchange in a particular encoding, for example if the data goes through some type of gateway or other application that converts the data to UTF-8, but does not change the encoding value in the XML declaration.

In this example, the XML input data will always be interpreted as UTF-8 data, regardless of what encoding information is provided within the data.



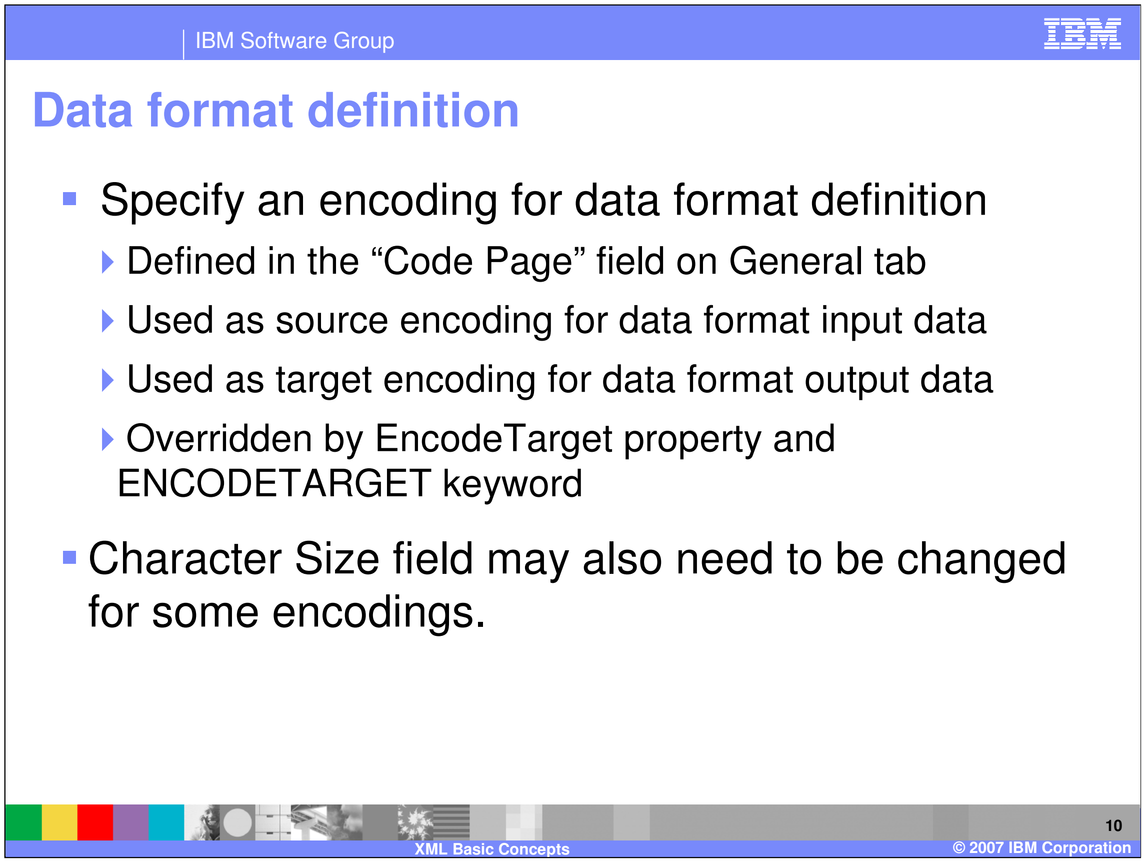
ENCODETARGET command keyword

- ENCODETARGET keyword
  - Used on the TRANSFORM command
  - Forces output data to be written using specified encoding
  - Overrides EncodeTarget property, data format definition
- Example:

```
PERFORM TRANSFORM WHERE INFILE(XMLFILE) OUTFILE(OUTFILE)
      SYNTAX(X) ENCODETARGET(UTF-8)
```

The ENCODETARGET keyword can also be specified on the TRANSFORM command, but it controls the encoding used to write the output data. This will override the the EncodeTarget property if the property is set in the map, or the code page specified in the data format definition. If you always want the output to use a particular encoding, this may be more convenient than setting the property in the map.

In this example, the output data will always be written as UTF-8 data.



IBM Software Group

Data format definition

- Specify an encoding for data format definition
  - Defined in the “Code Page” field on General tab
  - Used as source encoding for data format input data
  - Used as target encoding for data format output data
  - Overridden by EncodeTarget property and ENCODETARGET keyword
- Character Size field may also need to be changed for some encodings.

10

XML Basic Concepts

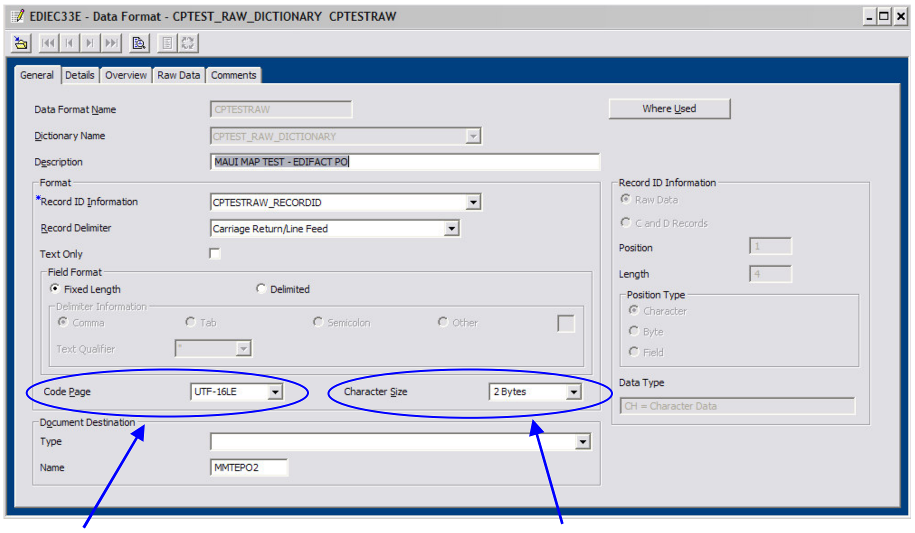
© 2007 IBM Corporation

You can specify a particular code page or encoding to be used for a particular data format. This is defined in the “Code Page” field on the Data Format General tab.

If this is defined, it is used as the source encoding when that data format is used as input data, and also as the target encoding when that data format is used as output data. You can override this value using the ENCODETARGET keyword or the EncodeTarget property.

If you specify a double-byte (16-bit) encoding such as UTF-16, you should also change the Character Size field. This allows WebSphere Data Interchange to correctly calculate fixed-format record offsets and lengths in terms of characters. For example, if a field contains 10 UTF-16 characters, it would actually contain 20 bytes, since each character is encoded as two bytes.

Example: Code page defined for data format



EDIEC33E - Data Format - CPTEST_RAW_DICTIONARY CPTESTRAW

General | Details | Overview | Raw Data | Comments

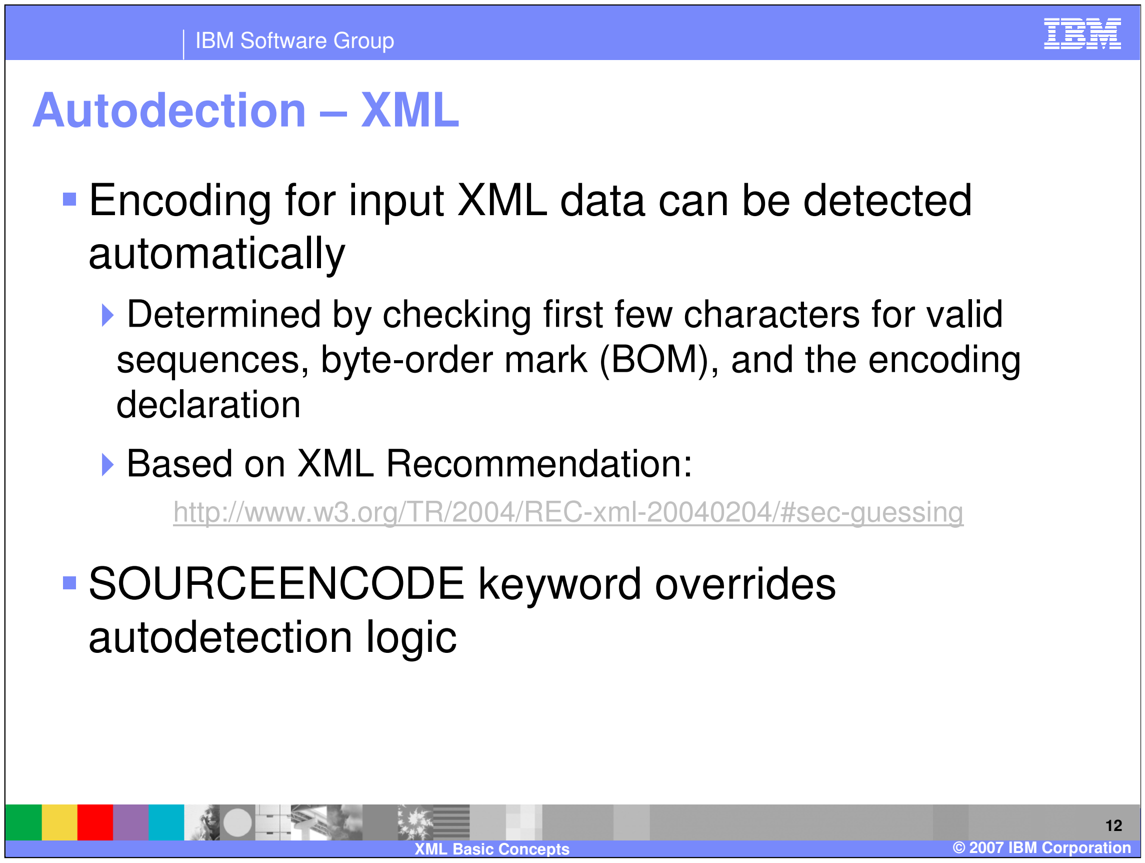
Data Format Name	CPTESTRAW
Dictionary Name	CPTEST_RAW_DICTIONARY
Description	MAUI MAP TEST - EDIFACT PO
*Record ID Information	CPTESTRAW_RECORDID
Record Delimiter	Carriage Return/Line Feed
Text Only	☐
Field Format	[✓ Fixed Length] Delimited
Code Page	UTF-16LE
Character Size	2 Bytes
Document Destination Type	
Document Destination Name	MMTEPO2
Record ID Information	[✓ Raw Data] C and D Records
Position	1
Length	4
Position Type	[✓ Character] Byte Field
Data Type	CH = Character Data

Code page to use

Size of each character

Here is an example of a data format definition that uses UTF-16LE as it's encoding. Since each UTF-16LE character is two bytes long, the Character Size field also needed to be changed.

If you are using an encoding such as UTF-8 where the character size varies, you should generally set the Character Size field to 1. Note that these types of encodings may be difficult to use with fixed-format records, since the record and field sizes contain fixed byte lengths.

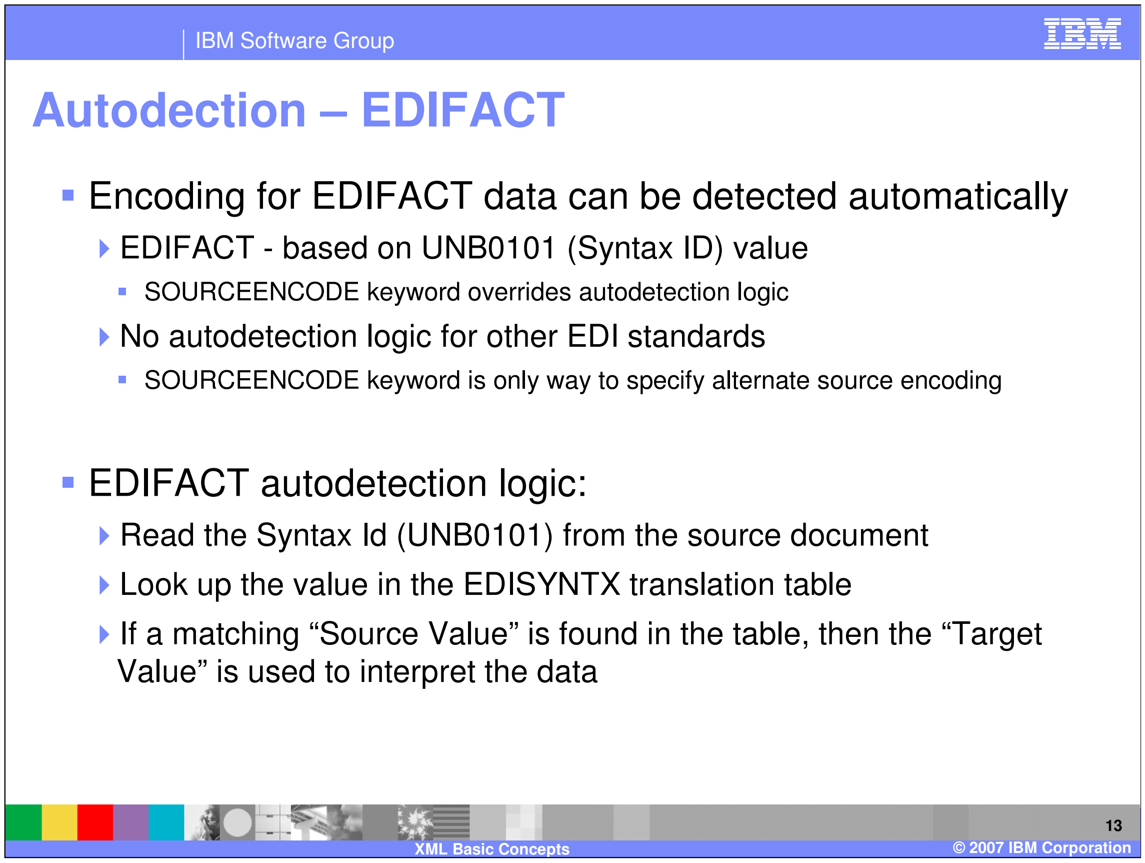


Autodection – XML

- Encoding for input XML data can be detected automatically
  - Determined by checking first few characters for valid sequences, byte-order mark (BOM), and the encoding declaration
  - Based on XML Recommendation:
    http://www.w3.org/TR/2004/REC-xml-20040204/#sec-guessing
- SOURCEENCODE keyword overrides autodetection logic

For XML input data, WebSphere Data Interchange can automatically determine the encoding by checking the first characters of data. It follows the rules described by the XML recommendation to determine the encoding based on whether it finds a byte-order mark (BOM), certain other valid sequences, and/or an encoding declaration. The details of this logic are described in the XML Recommendation, Appendix F Autodetection of Character Encodings.

As mentioned earlier, if the SOURCEENCODE keyword is specified it will override this autodetection logic.



Autodection – EDIFACT

- Encoding for EDIFACT data can be detected automatically
  - EDIFACT - based on UNB0101 (Syntax ID) value
    - SOURCEENCODE keyword overrides autodetection logic
  - No autodetection logic for other EDI standards
    - SOURCEENCODE keyword is only way to specify alternate source encoding

- EDIFACT autodetection logic:
  - Read the Syntax Id (UNB0101) from 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

The encoding for EDIFACT data, but not other EDI standards, can be determined from the data.

For EDIFACT data, the UNB0101 data element (Syntax Id) is used to determine the encoding. As usual, the SOURCEENCODE keyword overrides the autodetection logic. For other EDI standards, there is no autodetection capability, so the SOURCEENCODE keyword is the only way to specify an alternate source encoding.

For EDIFACT data, the following logic is used to determine the source encoding:

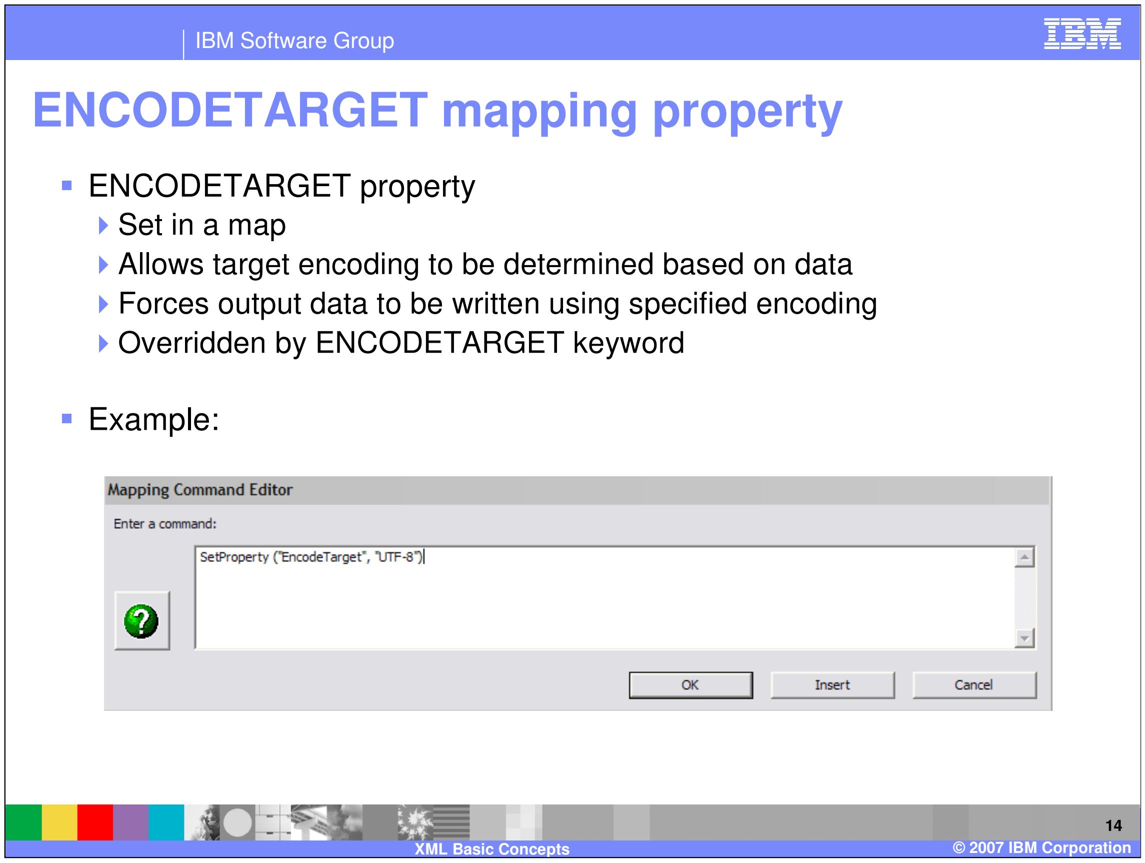
•Read the Syntax Id (UNB0101) from the source document.

•Look up this 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.

For example, if you wanted the source encoding to be “iso-8859-2” when the EDIFACT Syntax Id is “UNOD”, you would create an entry in the EDISYNTX table that had “UNOD” as the “Source Value”, and “iso-8859-2” as the target value.

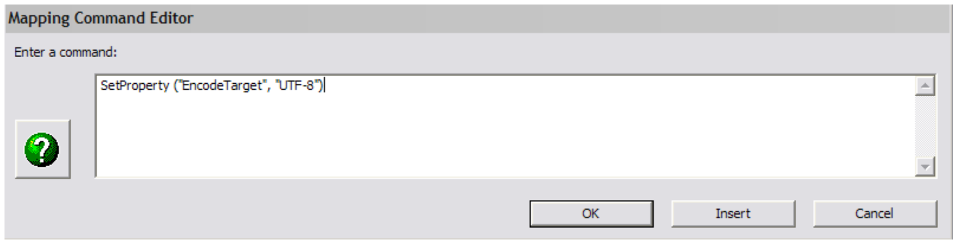
If the EDISYNTX table is not defined, or if it does not contain a “Source Value” that matches the Syntax id, then the default local code page is used.



IBM Software Group

ENCODETARGET mapping property

- ENCODETARGET property
  - Set in a map
  - Allows target encoding to be determined based on data
  - Forces output data to be written using specified encoding
  - Overridden by ENCODETARGET keyword
- Example:



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

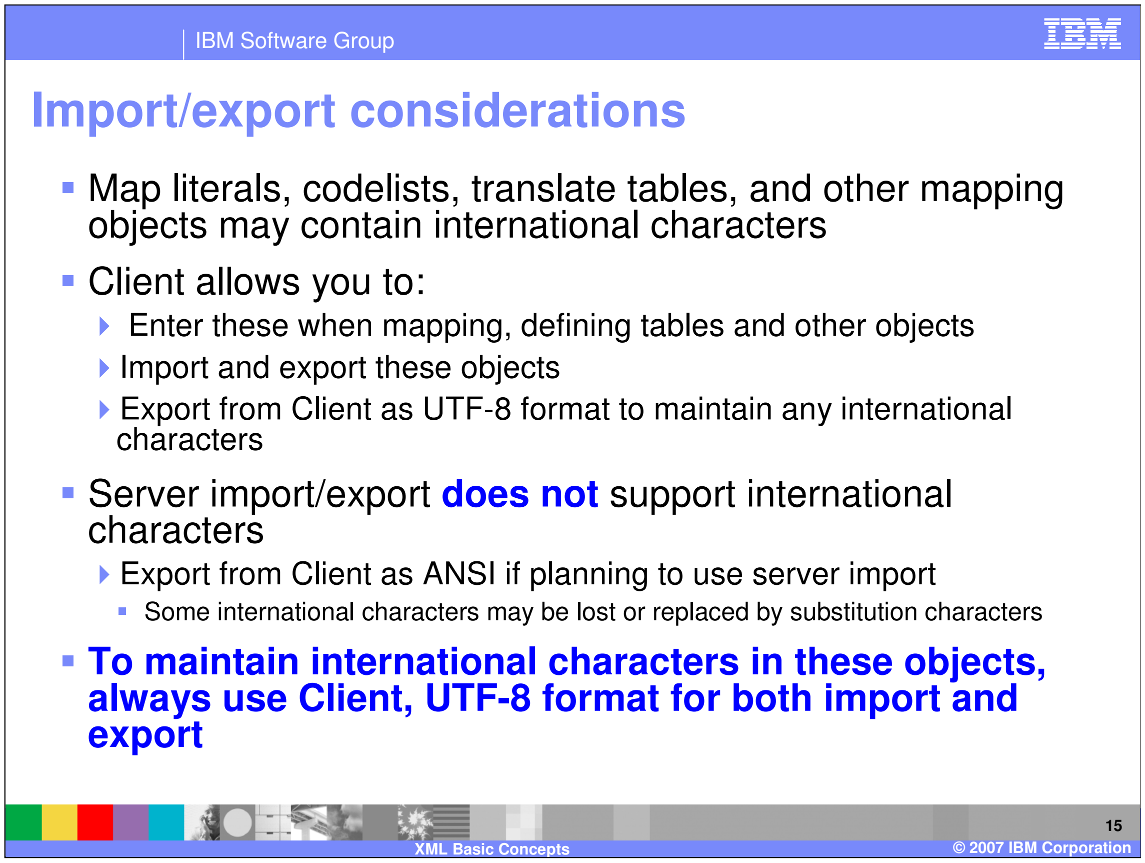
14

XML Basic Concepts

© 2007 IBM Corporation

The EncodeTarget property can be set within a Data Transformation map to control the encoding used to write the output data. By specifying this in the map instead of using the ENCODETARGET keyword, you can determine the encoding based on information within the data. For example, you may want to encode output data for some trading partners using UTF-8 because they are based in other countries, but just use the default codepage for other customers that are not able to process the UTF-8 data. In this example, the output data will be written as UTF-8 if this command is executed.

If the ENCODETARGET keyword is specified, it will override the property that is set in the map.


IBM Software Group

Import/export considerations

- Map literals, codelists, translate tables, and other mapping objects may contain international characters
- Client allows you to:
  - Enter these when mapping, defining tables and other objects
  - Import and export these objects
  - Export from Client as UTF-8 format to maintain any international characters
- Server import/export **does not** support international characters
  - Export from Client as ANSI if planning to use server import
    - Some international characters may be lost or replaced by substitution characters
- **To maintain international characters in these objects, always use Client, UTF-8 format for both import and export**

15

XML Basic Concepts

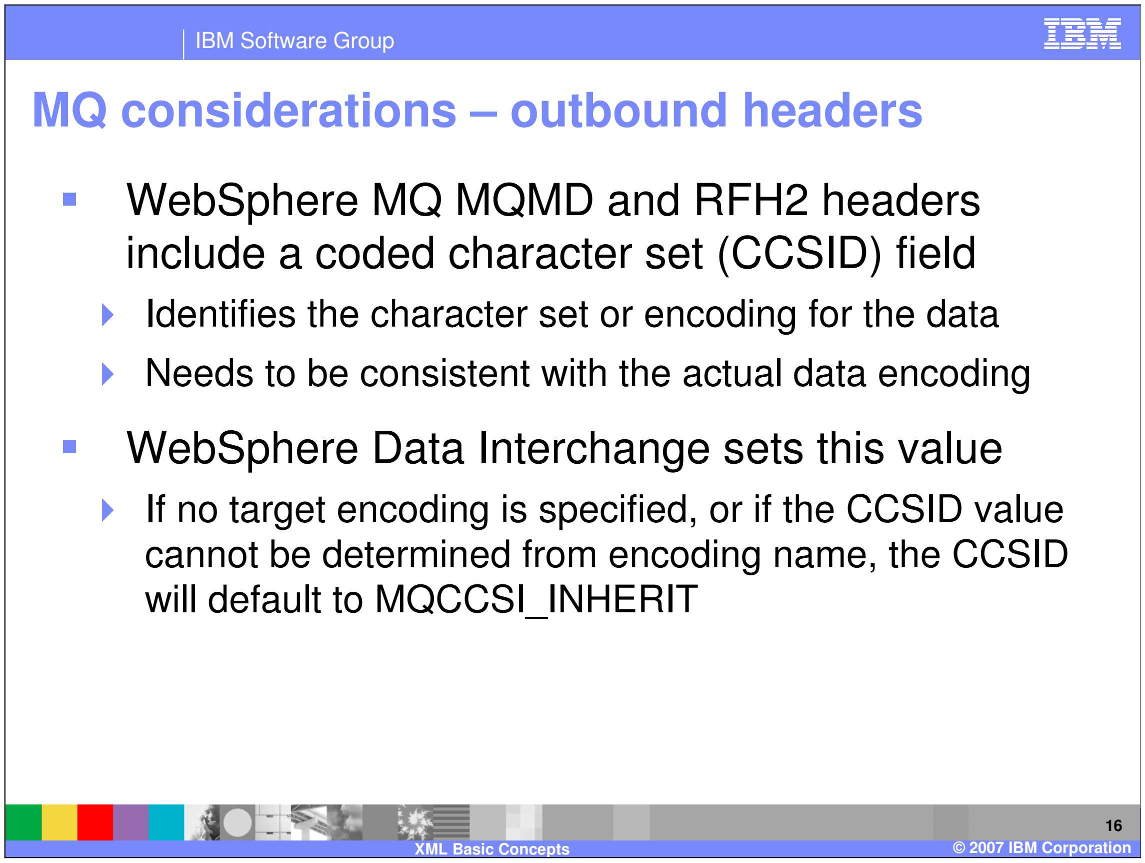
© 2007 IBM Corporation

You may wish to use international characters as part of map literals, codelists, translate tables, and other mapping objects. For example, you may want to assign a value that contains special characters into an output field, or define certain special characters as valid values for a field. WebSphere Data Interchange Client allows you to enter international characters in these objects just like you would any other characters. However, you need to be careful when exporting and importing these objects.

The Client can export data in either UTF-8 format or in ANSI format. UTF-8 format is a Unicode encoding, so any character can be represented using this format. If your objects contain international characters that cannot be represented using the default system codepage, you should export the mapping objects in UTF-8 format to make sure you do not lose these special characters.

ANSI format is the local default codepage for the Windows system where the Client is running. Files exported as ANSI can be transferred to the Server (and translated to EBCDIC if the Server is running on z/OS) and used for a PERFORM IMPORT command. WebSphere Data Interchange Server does not support import of UTF-8 import files. The drawback to using ANSI format is that any international characters may not be transferred correctly.

Therefore, it is important that if you are using international characters in your maps and other related database objects, you should always export and import these objects using the Client, and in UTF-8 format.


IBM Software Group

MQ considerations – outbound headers

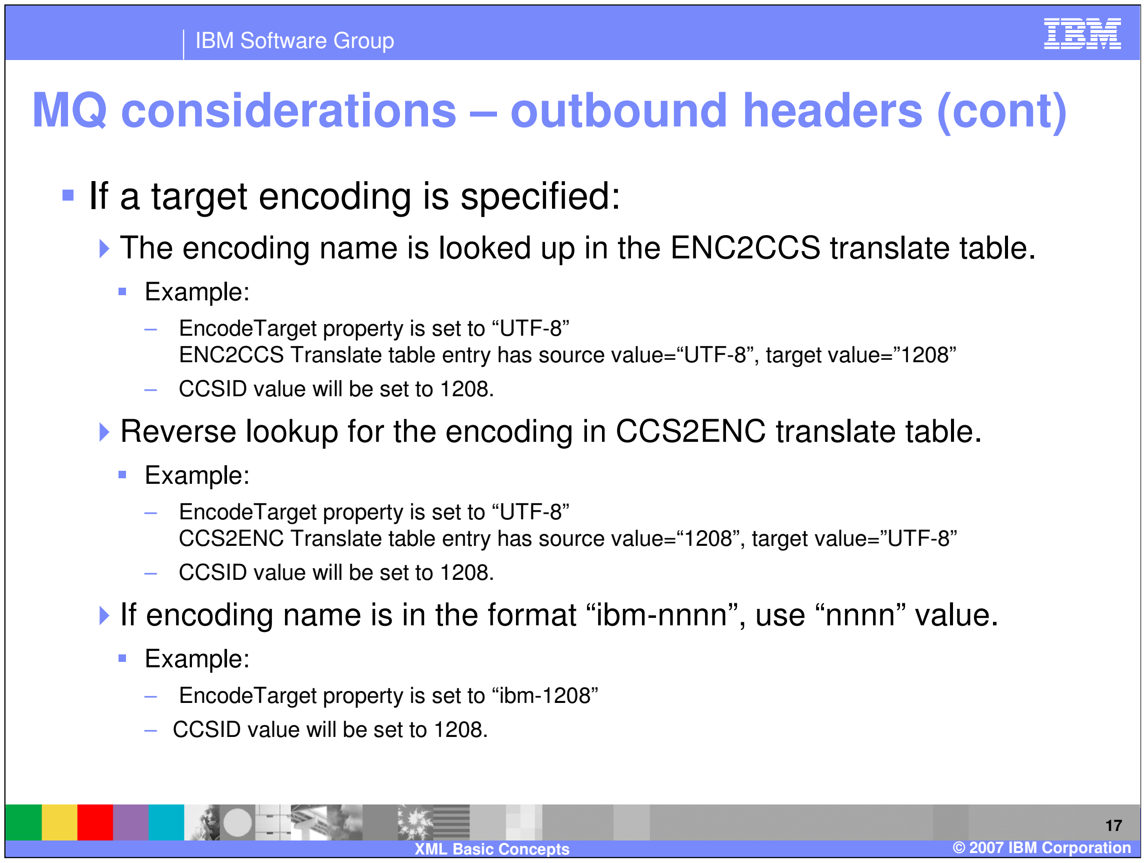
- WebSphere MQ MQMD and RFH2 headers include a coded character set (CCSID) field
  - Identifies the character set or encoding for the data
  - Needs to be consistent with the actual data encoding
- WebSphere Data Interchange sets this value
  - If no target encoding is specified, or if the CCSID value cannot be determined from encoding name, the CCSID will default to MQCCSI_INHERIT

16

XML Basic Concepts © 2007 IBM Corporation

When sending the output data to a WebSphere MQ queue, WebSphere Data Interchange also generates the MQMD header and optionally the RFH2 header along with the data. These headers include a coded character set id, or CCSID, which is an integer value that identifies the character set or encoding for the data. The value in this field should be consistent with the character set or encoding actually used for the data, or the data may be interpreted incorrectly by the application that receives the data.

WebSphere Data Interchange uses the target encoding to help determine what value it should use to set the CCSID field. If no target encoding is specified, or the CCSID cannot be determined from the target encoding name, the default value MQCCSI_INHERIT is used to indicate that the data uses the same character set as the queue manager.


IBM Software Group

MQ considerations – outbound headers (cont)

- If a target encoding is specified:
  - The encoding name is looked up in the ENC2CCS translate table.
    - Example:
      - EncodeTarget property is set to “UTF-8”
        ENC2CCS Translate table entry has source value=“UTF-8”, target value=”1208”
      - CCSID value will be set to 1208.
  - Reverse lookup for the encoding in CCS2ENC translate table.
    - Example:
      - EncodeTarget property is set to “UTF-8”
        CCS2ENC Translate table entry has source value=“1208”, target value=”UTF-8”
      - CCSID value will be set to 1208.
  - If encoding name is in the format “ibm-nnnn”, use “nnnn” value.
    - Example:
      - EncodeTarget property is set to “ibm-1208”
      - CCSID value will be set to 1208.

17

XML Basic Concepts

© 2007 IBM Corporation

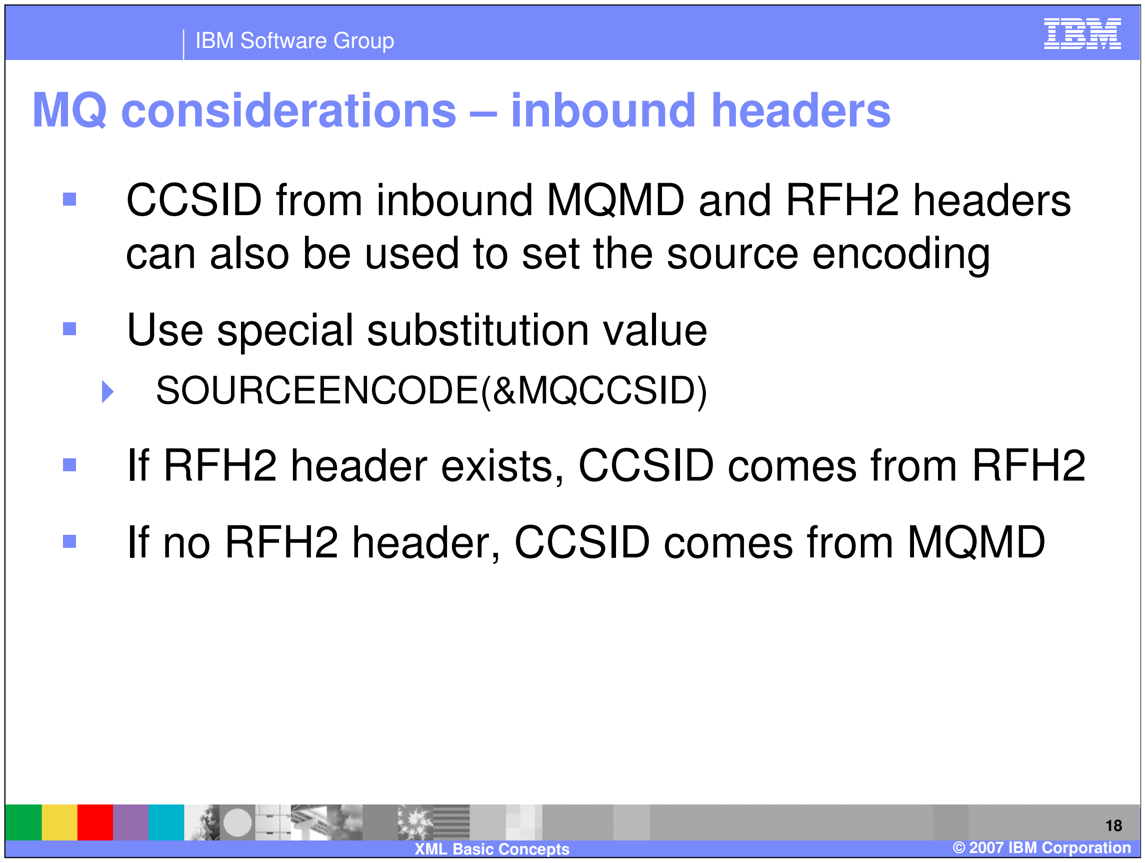
If a target encoding is specified using either the EncodeTarget property or the ENCODETARGET keyword, then the CCSID is determined as follows:

•First, the encoding name is looked up in the ENC2CCS translate table. If it is found, the translated value is used as the CCSID.

•If the CCSID is not found in the first step, a reverse lookup will be done for the encoding in CCS2ENC translate table. If this is found, the translated value, which is the source value in the table entry, is used as the CCSID.

•If the CCSID is still not found, but the encoding name is in the format “ibm-nnnn”, the numeric part of the value in the encoding name is used as the CCSID.

The examples on this slide help illustrate how each of these steps might be used. If none of these steps are able to convert the target encoding name to a CCSID value, then the CCSID value in the MQ headers are set to the default values.

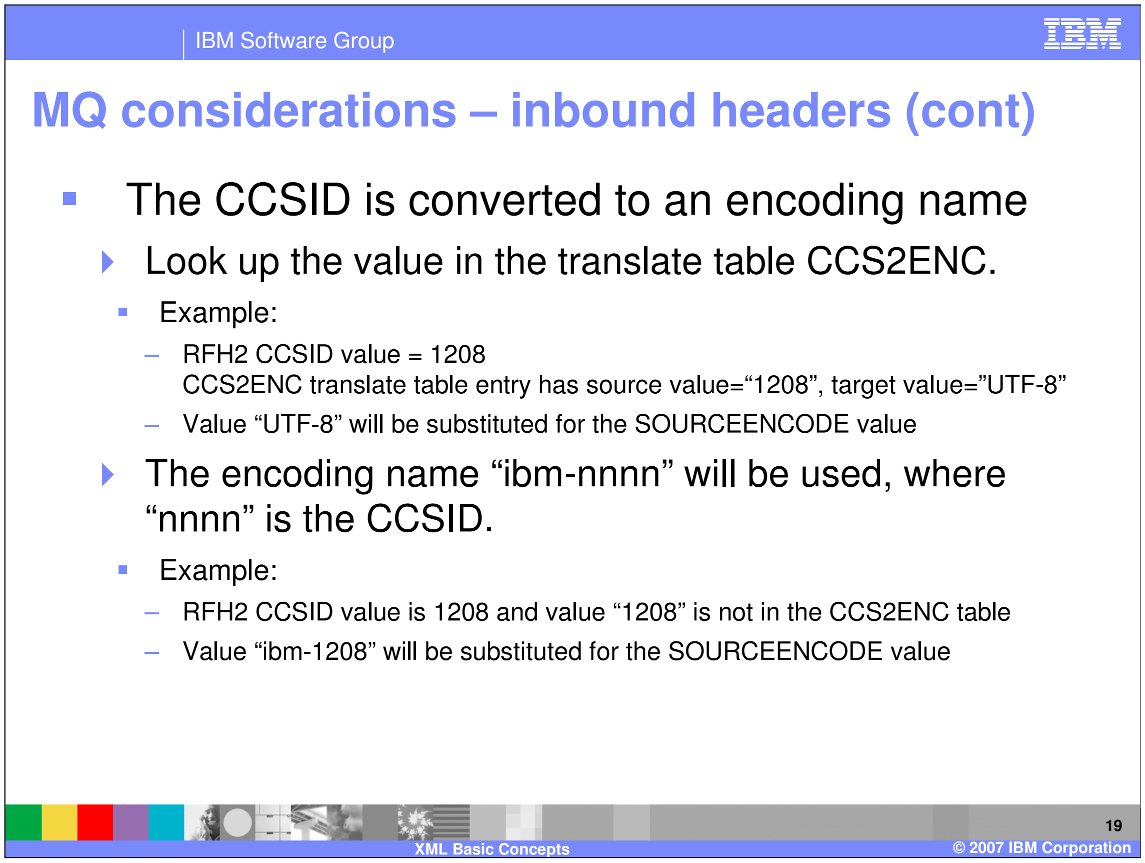


MQ considerations – inbound headers

- CCSID from inbound MQMD and RFH2 headers can also be used to set the source encoding
- Use special substitution value
  - SOURCEENCODE(&MQCCSID)
- If RFH2 header exists, CCSID comes from RFH2
- If no RFH2 header, CCSID comes from MQMD

The CCSID in the MQMD or RFH2 header can also be used to determine the source encoding when processing data that was received from a WebSphere MQ queue. To do this, the special value &MQCCSID is used for the SOURCEENCODE keyword.

If the data has an RFH2 header, the CCSID from the RFH2 is used to determine the source encoding. If not, the CCSID from the MQMD header is used.



MQ considerations – inbound headers (cont)

- The CCSID is converted to an encoding name
  - Look up the value in the translate table CCS2ENC.
    - Example:
      - RFH2 CCSID value = 1208
        CCS2ENC translate table entry has source value="1208", target value="UTF-8"
      - Value "UTF-8" will be substituted for the SOURCEENCODE value
  - The encoding name "ibm-nnnn" will be used, where "nnnn" is the CCSID.
    - Example:
      - RFH2 CCSID value is 1208 and value "1208" is not in the CCS2ENC table
      - Value "ibm-1208" will be substituted for the SOURCEENCODE value

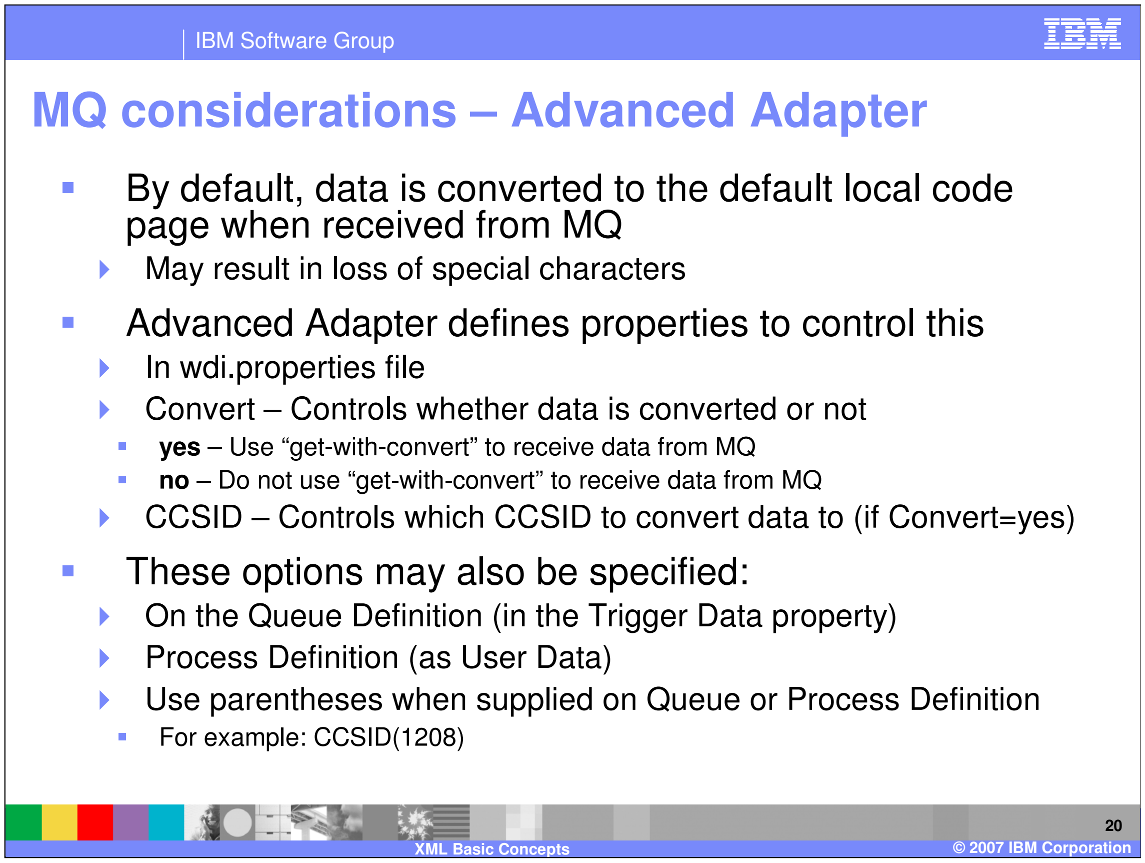
Once the CCSID is read from the appropriate header, it is converted to an encoding name. The encoding name is determined as follows:

•First, the CCSID value is looked up in the CCS2ENC translate table. If it is found, the translated value is used as the encoding name.

•If the CCSID is was not found in the table, the encoding name is set to "ibm-nnnn", where the numeric CCSID value is used for the nnnn part of the name.

There are examples on this slide to help illustrate each of these steps.

The converted value will be used as the SOURCEENCODE value and an informational message will be issued to indicate which encoding name was used.


IBM Software Group

MQ considerations – Advanced Adapter

- By default, data is converted to the default local code page when received from MQ
  - May result in loss of special characters
- Advanced Adapter defines properties to control this
  - In wdi.properties file
  - Convert – Controls whether data is converted or not
    - **yes** – Use “get-with-convert” to receive data from MQ
    - **no** – Do not use “get-with-convert” to receive data from MQ
  - CCSID – Controls which CCSID to convert data to (if Convert=yes)
- These options may also be specified:
  - On the Queue Definition (in the Trigger Data property)
  - Process Definition (as User Data)
  - Use parentheses when supplied on Queue or Process Definition
    - For example: CCSID(1208)

20

XML Basic Concepts © 2007 IBM Corporation

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 special characters, but the queue manager CCSID is for US-ASCII, then the special characters may be lost.

The WDI Advanced Adapter allows you to control this data conversion through the "wdi.properties" file. Two properties are used to control this:

· The “Convert” property determines if WebSphere MQ conversion tables should be used to convert the data when it is received, and it is set to either "yes" or "no." The default is "Convert=yes."

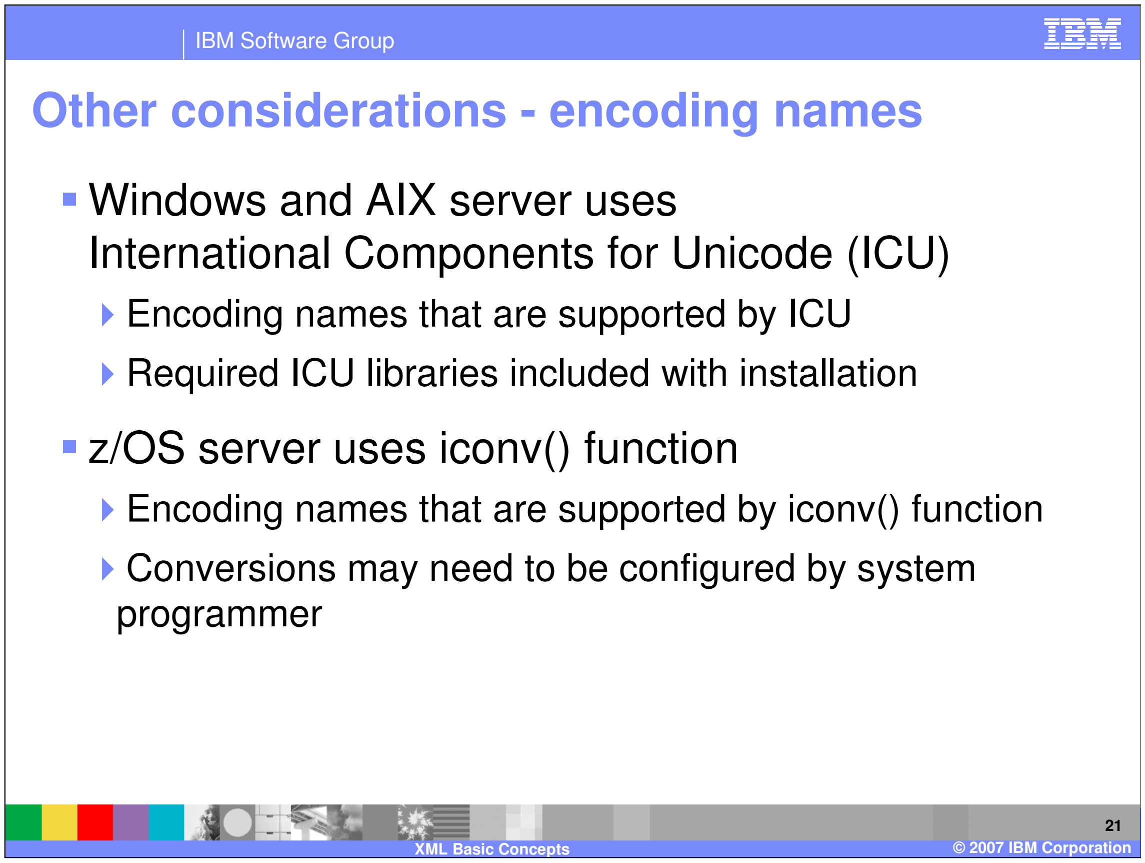
· The "CCSID" option controls which CCSID you want to the data to be converted to, and overrides the typical conversion to the Queue Manager's Local Code Page. It has a numeric argument that corresponds to the Coded Character Sets as defined by IBM.

In addition to their use in the wdi.properties file, these options may alternatively be specified:

•On the WebSphere MQ Queue definition in the Trigger Data property

•In the WebSphere MQ Process Definition as User Data.

When specified as part of the WebSphere MQ objects, the values should be in parenthesis

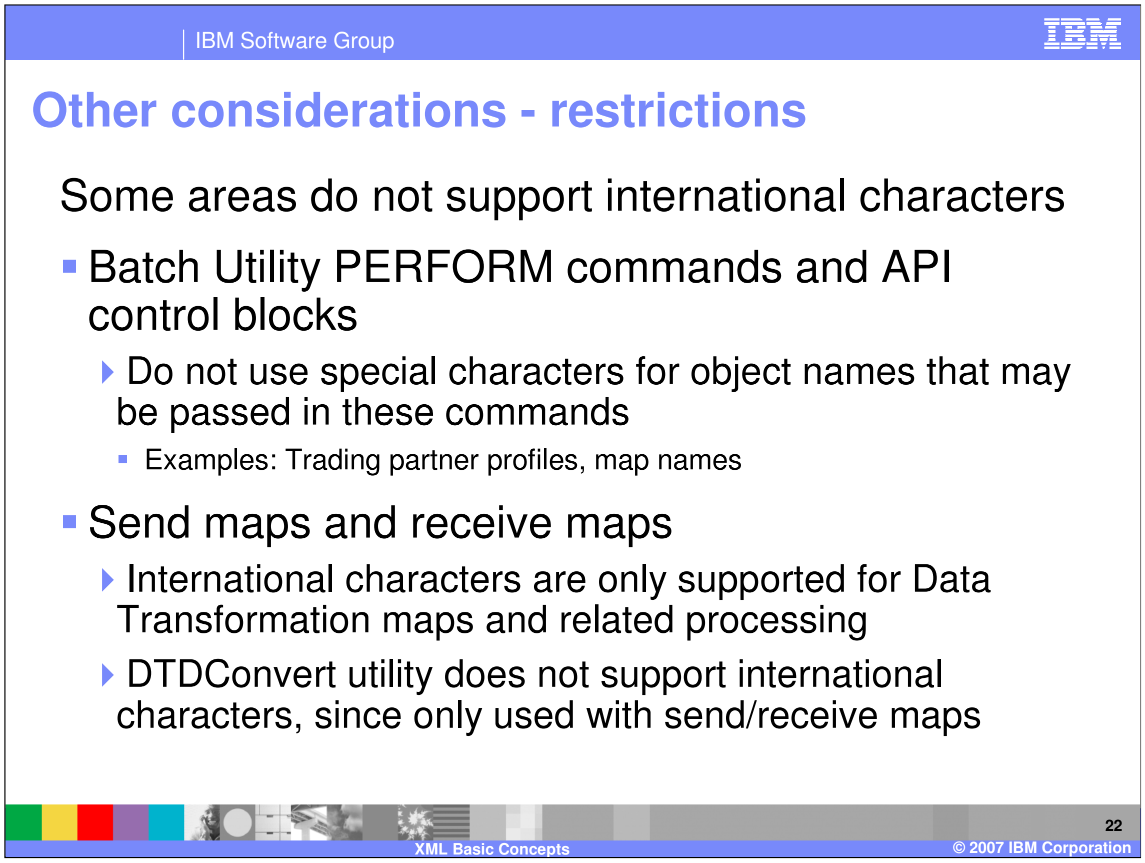


Other considerations - encoding names

- Windows and AIX server uses International Components for Unicode (ICU)
  - Encoding names that are supported by ICU
  - Required ICU libraries included with installation
- z/OS server uses iconv() function
  - Encoding names that are supported by iconv() function
  - Conversions may need to be configured by system programmer

On Windows and AIX, Websphere Data Interchange server uses International Components for Unicode, often called ICU, to convert data from one encoding to another. This set of libraries supports numerous different encodings. Encoding names supported by ICU are generally also supported by WebSphere Data Interchange on Windows and AIX. The required ICU libraries are installed as part of the WebSphere Data Interchange installation.

On z/OS, the iconv() function is used for most conversions. Encoding names recognized by the iconv() function are generally supported by WebSphere Data Interchange on z/OS. Your system programmer may need to configure some conversions in order for the iconv() function and the z/OS operating system to recognize them.



Other considerations - restrictions

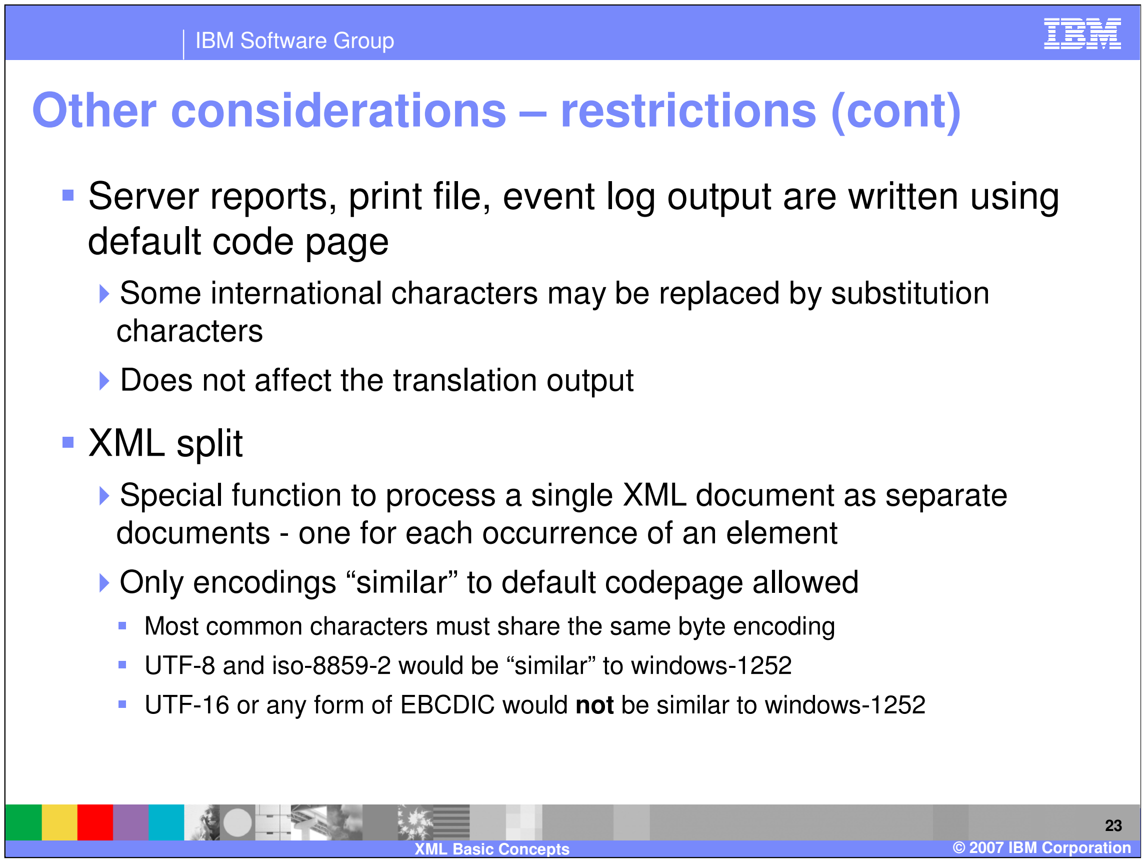
Some areas do not support international characters

- Batch Utility PERFORM commands and API control blocks
  - Do not use special characters for object names that may be passed in these commands
    - Examples: Trading partner profiles, map names
- Send maps and receive maps
  - International characters are only supported for Data Transformation maps and related processing
  - DTDConvert utility does not support international characters, since only used with send/receive maps

There are some areas of WebSphere Data Interchange that still read and write information using the default local codepage. These are generally areas where reading and writing the data in a Unicode format such as UTF-8 or UTF-16 would make the information difficult to use – especially in a z/OS environment where most editors and viewers could not display the data. In some cases such as send and receive map processing, the function is primarily for migration from previous releases, and it was not considered practical to make the changes required to handle other character encodings.

One area that does not support international characters is the Batch Utility PERFORM command processing, and similarly the API control blocks that are used to call the translator using the API. The information in the PERFORM commands and keywords values, and in the API control blocks, is all specified in the default system code page. This means that any characters outside this code page cannot be passed as parameters for the commands and API calls. Because of this, it is a good practice **not** to use these characters for object names such as trading partner profile names, map names, and so forth.

The international characters and different encodings are only supported for the Data Transformation maps and related processing. Send map and receive map processing does not support international characters. Similarly, since the DTDConvert utility is used only for doing XML translation with send and receive maps, the DTDConvert utility does not support international characters either.


IBM Software Group

Other considerations – restrictions (cont)

- Server reports, print file, event log output are written using default code page
  - Some international characters may be replaced by substitution characters
  - Does not affect the translation output
- XML split
  - Special function to process a single XML document as separate documents - one for each occurrence of an element
  - Only encodings “similar” to default codepage allowed
    - Most common characters must share the same byte encoding
    - UTF-8 and iso-8859-2 would be “similar” to windows-1252
    - UTF-16 or any form of EBCDIC would **not** be similar to windows-1252

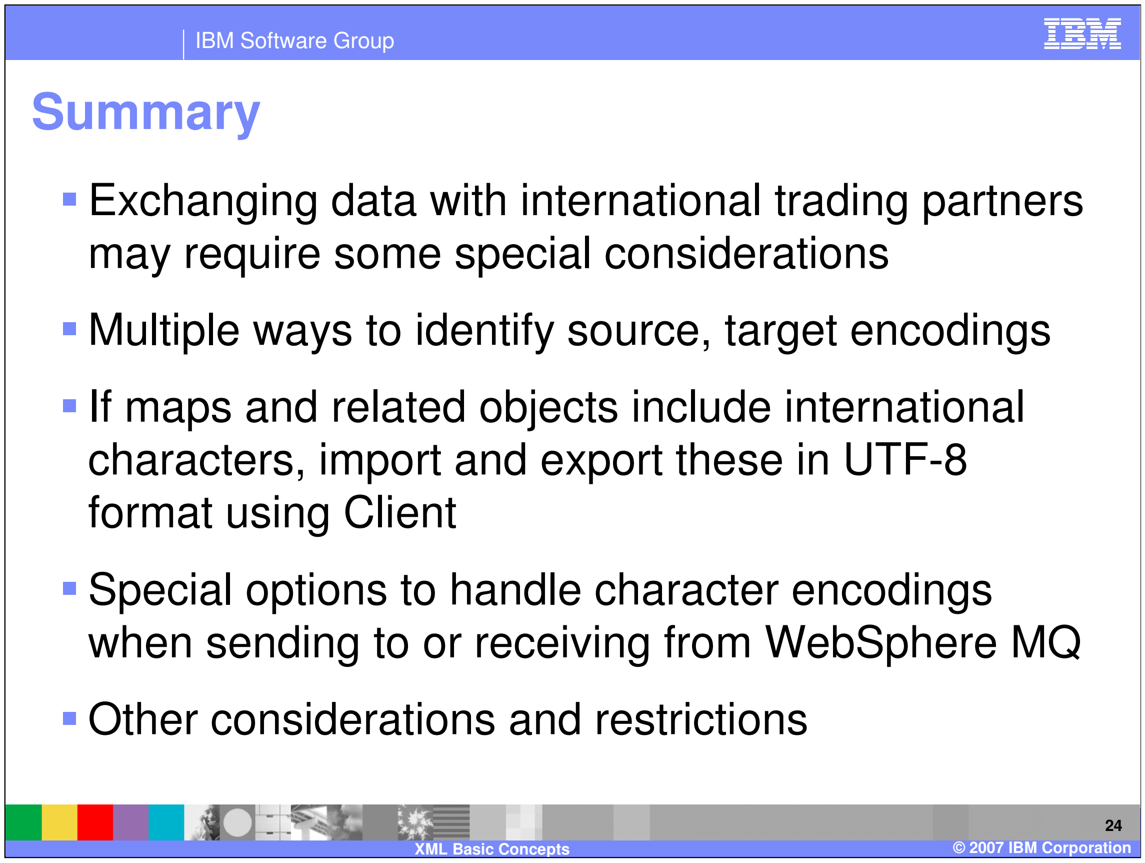
23

XML Basic Concepts © 2007 IBM Corporation

Server reports, print file (PRTFILE) messages, and event log output are all written using the default local codepage. If the data contains international characters that are inserted in these reports or messages, but cannot be represented in the default code page, substitution characters may appear in place of those characters. Note that this does not affect the output image from the translation, but only the text of the reports or messages.

Finally, the “XML split” function has some restrictions on the character encodings that it can handle. This function allows XML data to be split into multiple XML documents – one document for each occurrence of a particular repeating element. When using this function, the data can use a character encoding that is “similar” to the default code page. That is, most common characters use the same byte encoding as the default code page. For example, encodings such as UTF-8 and iso-8859-2 would be generally be similar to windows-1252, since they use the same byte encoding for lowercase characters a-z, uppercase A-Z, digits 0-9, and most punctuation. (If special characters in the upper range of windows-1252 were used in the elements used to split the document, the encodings might no longer be considered similar.)

Encodings where the common characters do **not** share the same byte values would not be considered similar. For example, UTF-16 or any for of EBCDIC would not be supported if the default code page was windows-1252.


IBM Software Group

Summary

- Exchanging data with international trading partners may require some special considerations
- Multiple ways to identify source, target encodings
- If maps and related objects include international characters, import and export these in UTF-8 format using Client
- Special options to handle character encodings when sending to or receiving from WebSphere MQ
- Other considerations and restrictions

24

XML Basic Concepts © 2007 IBM Corporation

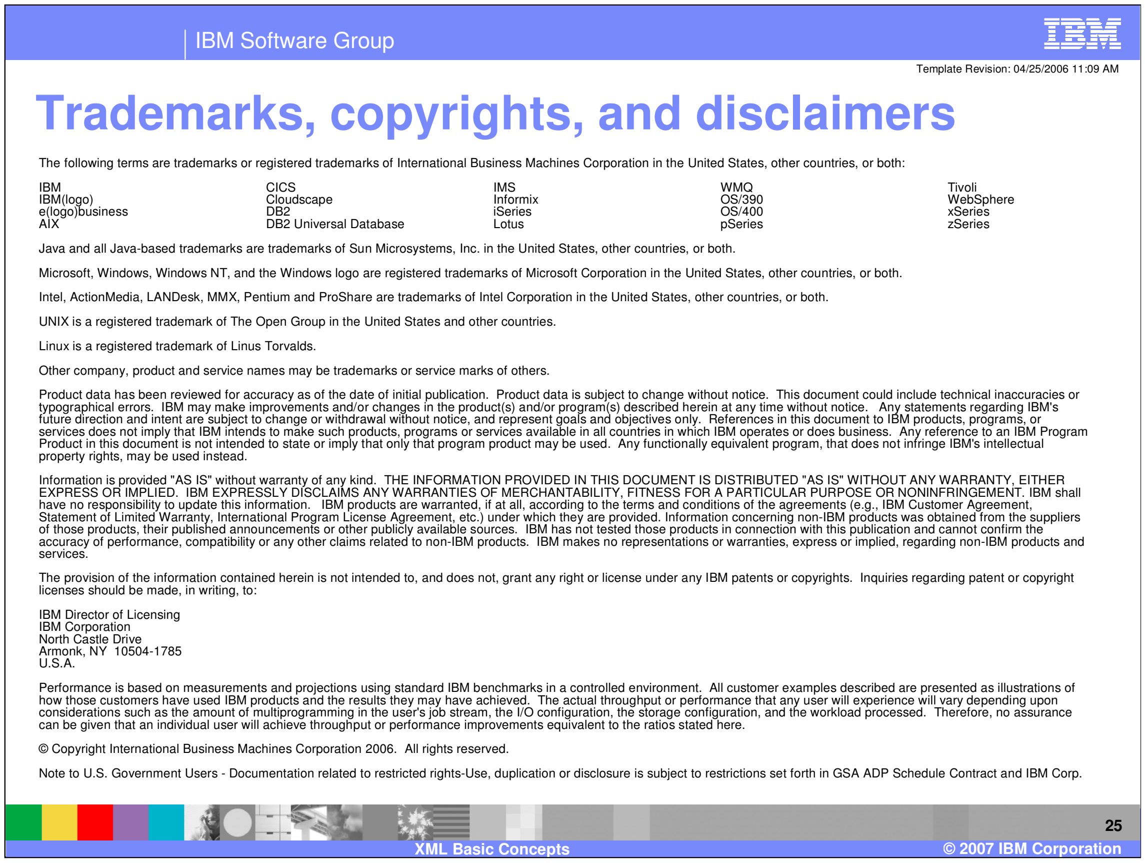
In summary, there are some extra requirements you may need to consider when exchanging data with trading partners in other countries – especially if their native language is not English. Otherwise, some of the characters in their data may be lost or corrupted.

WebSphere Data Interchange allows multiple ways to identify the source and target encodings for the data.

If maps and other related objects such as code lists and translation tables contain international characters, these should be imported and exported in UTF-8 format using the Client. Server import/export does not support characters outside of the default code page.

There are also some special options that you can use to handle different characters encodings when sending data to or receiving data from WebSphere MQ.

Finally, there are a few other miscellaneous considerations and restrictions that you should be aware of when processing different character encodings with WebSphere Data Interchange.



Template Revision: 04/25/2006 11:09 AM

Trademarks, copyrights, and disclaimers