

Data Integrator

Encoding Reference

Pervasive Software, Inc.
12365-B Riata Trace Parkway
Austin, Texas 78727 USA

Telephone: 888.296.5969 or 512.231.6000

Fax: 512.231.6010

Email: info@pervasiveintegration.com

Web: <http://www.pervasiveintegration.com>



See copyrights.txt in the product installation directory for information on third-party and open source software components.

Encoding Reference
October 2010

Contents

1	Specifying Encoding and Unicode	1-1
	<i>What Is Encoding and Unicode and How Is it Supported?</i>	
	Understanding Unicode and Other Encoding Types	1-2
	Unicode Defined	1-2
	Non-Unicode Encoding	1-3
	Integration Platform Support	1-3
	Unicode and Encoding Checklist	1-4
	Installing Language Sets	1-4
	Installing Character Sets	1-4
	Installing Font Sets	1-5
	Customizing Character Mappings	1-6
	International Components for Unicode	1-6
2	Procedures and Examples	2-1
	<i>Configuration and Design Considerations</i>	
	Determining Which Connector to Use	2-2
	Flat File Connectors	2-2
	Unicode or ASCII	2-2
	Character Encoding	2-2
	Unicode Source Data	2-2
	Unicode Target Data	2-2
	XML Connector	2-3
	Database Connectors	2-3
	Setting the Correct Encoding Options During Installation	2-3
	ODBC Driver Support	2-3
	Oracle Connector	2-3
	Sybase Connector	2-3
	SQL Server Connector	2-3
	CRM Connectors	2-3
	Components	2-4
	Encoding RIFL Functions and Objects	2-5
	Encoding Functions	2-5
	File Functions	2-5
	File Function Examples	2-5
	RIFL Objects	2-6
	DJMessage Objects	2-6
	Configuring a Sybase Database for Encoding	2-7
	Configuring an Oracle Database for Encoding	2-8
	Customer Example	2-8
A	Appendix	A-1
	<i>Supplemental Information</i>	

Contents

Troubleshooting Encoding Issues	A-2
Unicode 4.0 Limitations	A-4
Data Browser	A-4
RIFL Script Editor	A-4
Data Parser	A-4
Unicode 4.0 Character Encodings	A-5
Customizable ICU Encodings	A-9
Customizing Non-ICU Japanese Mappings	A-12
Encoding Constant Values	A-13
Character Encoding Constants	A-13
Sending Text and Binary Email	A-19
Byte Order Marks in Microsoft Windows	A-20

About This Manual

This manual leads you through encoding concepts, configuration, and procedures for Data Integrator.

It is intended for developers and consultants who want to use encoding in connectors, function scripts, and object parameters.

You must be familiar with normal setup procedures in a Windows or Unix environment. Experience with the basic operation of the integration platform tool set is also helpful.

Specifying Encoding and Unicode

chapter

1

What Is Encoding and Unicode and How Is it Supported?

This chapter includes the following topics:

- “Understanding Unicode and Other Encoding Types” on page 1-2
- “Unicode and Encoding Checklist” on page 1-4
- “Customizing Character Mappings” on page 1-6

Understanding Unicode and Other Encoding Types

Unicode Defined

Unicode is an international character set. Like other character sets such as American Standard Code for Information Interchange (ASCII), the Unicode character set provides a standard correspondence between the binary numbers that computers understand, and the letters, digits, and punctuation that people understand.

Unlike ASCII, however, Unicode provides a code for every character in nearly every language in the world. This task requires more than the 256 characters available in ASCII. ASCII is based on the 8-bit character set, while Unicode uses 16-bit characters as the default.

Unicode characters are most commonly referred by their 4-digit hexadecimal representations (0000 to FFFF). The numbers 0 (0000) to 128 (007F) correspond exactly to their ASCII counterparts. The correspondence between the integer values and the actual characters may be found at [Unicode's website](#).

Unicode includes the Latin alphabet used for English, the Cyrillic alphabet used for Russian, the Greek, Hebrew, and Arabic alphabets, and other alphabets used in Europe, Africa, and Asia, such as Japanese kana, Korean hangul, and Chinese bopomofo.

Much of the Unicode standard includes thousands of unified character codes for Chinese, Japanese, and Korean ideographs. Adopted as an international standard in 1992, Unicode was originally a "double-byte," or 16-digit, binary number code that could represent up to 65,536 items.

UTF Format Encoding in Applications

No longer limited to 16 bits, Unicode can currently represent about one million code positions using three encoding forms called Unicode Transformation Formats (UTF) as shown below:

UTF Format	Number of Bytes	Application
UTF-8	Consists of one-, two-, three-, and four-byte codes	Used in World Wide Web applications. Widely used because it is backwards compatible with ASCII, since all 128 US-ASCII characters have the same single-byte code points as they would in ASCII.
UTF-16	Consists of two- and four-byte codes	Also called a double-byte character set (DBCS). Used primarily for data storage and text processing. Developed for Japanese, Chinese and Korean languages.
UTF-32	Consists of four-byte codes	Used when character handling efficiency is important.

Non-Unicode Encoding

ASCII characters are the most commonly known non-Unicode characters. English language characters are often included in the first 128 code points (Hex 00-9F) in non-Unicode code pages. However, in the popular Japanese ShiftJIS character set, many of the first 128 code points are reserved as lead bytes and are not available for English characters.

In SQL Server 2000, the non-Unicode character data types are char, varchar, and text. These data types use the character representation scheme defined in single or double-byte code pages in SQL Server.

Integration Platform Support

The integration platform supports the Unicode 4.0 standard and also includes support for encoding non-Unicode data. This documentation leads you through the various locations where you can set encoding in connector properties, RIFL functions and objects.

Unicode and Encoding Checklist

When preparing to work with Unicode and other encoded data, there are several requirements to consider before you begin.

- Database configuration: Your server, client and database must be configured to use the correct character set.
- Field data types: The data types must support Unicode or your encoding type.
- Determine the encoding of your data files or tables. For instance, is the encoding UTF-8, ASCII, or UTF-16?
- Install drivers that support your data's encoding type.
- Keep in mind that Data Integrator supports the Unicode 4.0 standard.
- Install the following items that are required for your data format:
 - Language sets
 - Character sets
 - Font sets

For more information on installing these required items, see the sections below.

Installing Language Sets

You can leave your operating system default language setting in the Windows Regional and Language Options set to your native language. The transfer of data and the viewing of Unicode data characters is not affected by this setting.

Regardless of your default language setting, to view certain characters correctly, you will need to have those language sets, such as East Asian languages, installed.

Installing Character Sets

Character sets primarily affect your file format and how data is stored and transmitted, as well as string processing. Character sets do not resolve formatting issues, special input requirements, or display issues.

If you do not have the appropriate character sets installed on the client you are using to run Map Designer, you may see incorrect results when using OEM encoding.

In addition, you must have the appropriate Unicode character sets installed to set the encoding to UTF-8, UTF-16, and so on.

Installing Font Sets

Map Designer transformations are only concerned with data, so transformations should work correctly, regardless of the fonts or languages installed on your machine. However, to view data in the data browsers, you must install the appropriate font sets.

To change your font set, select **View > Preferences > Font**. After selecting the font, set the appropriate script setting for the font.

The following shows Map Designer font and script settings for a few languages:

Language	Grid Font Setting	Script Setting
Japanese	MS Gothic	Japanese
Korean	Batang	Hangul
Simplified Chinese	SimHei	Chinese GB2312
Traditional Chinese	MingLiU	Chinese Big5

Customizing Character Mappings

Support is provided for custom character mappings of standards such as International Components for Unicode (ICU) and non-ICU Japanese encodings.

International Components for Unicode

The integration platform uses International Components for Unicode 3.2 to convert internally between Unicode and most other source and target data encodings.

The following procedure shows how to customize these encoding conversions using mapping tables and a conversion utility to generate the ICU 3.2 character mappings.

The mapping tables have the file extension `.ucm`. The conversion utility is called `makeconv.exe`. Contact Engineering Support to receive a `.zip` archive that includes this `.exe` file.

➤ **To customize an ICU encoding**

The default for *InstallDir* is `C:\Program Files\InstallDir\Cosmos9`. In this location and its counterpart on Windows 7, you must run `makeconv.exe` as administrator.

- 1 Download the `.zip` archive for ICU customization and extract it to `InstallDir\Common\mappings`.
- 2 In the extracted mappings directory, find the file `makeconv.exe` and a folder called `tables` containing `.ucm` files.
- 3 Move `makeconv.exe` up a level to the `Common` directory.
- 4 In the `tables` folder, select the `.ucm` file to customize. For reference, see “Customizable ICU Encodings” on page A-9.
- 5 Copy the selected `.ucm` file to the `Common` directory.
- 6 Using [ICU encoding rules](#), open the `.ucm` file in a text editor, customize the character mappings, then save and close the file.
- 7 Open a command prompt in the `Common` directory and run:

```
makeconv mapping-table.ucm
```

where *mapping-table* is the name of the edited file.

Makeconv produces the file *mapping-table.cnv*.

- 8 Move the new .cnv file to the mappings directory.
- 9 Restart any designer applications.

Encoding conversions to and from Unicode are now handled based on the custom mappings.



Note A user-defined folder other than Common\mappings can be set in the cosmos.ini file by adding the following lines:

[Preferences]

EncMappingsPath=*pathname*

You can use two or more .cnv files with the same file name but alternate mappings by giving each file its own folder. To select the .cnv file to use, change the EncMappingsPath value.

See Also

“Customizable ICU Encodings” on page A-9

Procedures and Examples

chapter

2

Configuration and Design Considerations

The topics in this chapter include:

- “Determining Which Connector to Use” on page 2-2
- “Encoding RIFL Functions and Objects” on page 2-5
- “Configuring a Sybase Database for Encoding” on page 2-7
- “Configuring an Oracle Database for Encoding” on page 2-8

Determining Which Connector to Use

Flat File Connectors

Unicode or ASCII

When working with text files with special encoding characters, you should use a Unicode connector instead of an ASCII connector. Even if you do not have Unicode data, you have the option of setting the encoding property in the Unicode connectors. The encoding property options include non-Unicode and Unicode formats.

Character Encoding

Setting the correct character encoding is important. If you are not sure which encoding to use, select UTF-8. It is the most common Unicode scheme and handles most issues. If needed, you can also use ASCII encoding.

Unicode Source Data

When connecting a Unicode source data file with the Unicode connector, the encoding scheme should automatically change from OEM to the proper character set after the source file is specified. For instance, your file may specify UCS-2 when used as a source file. However, if the file does not have leading byte characters as described below, you may have to set the encoding scheme (to UCS-2 or UTF-16, for example).

Unicode Target Data

When outputting to a Unicode target data file with the Unicode connector, the encoding scheme defaults to OEM. You should change the encoding to reflect the correct Unicode character set such as UTF-8 or UTF-16. Note that choosing UCS-2 in Map Designer will provide the same results as choosing UTF-16. For the purposes of data exchange, UTF-16 and UCS-2 are the same. Both are 16-bit and have the same code unit representation.

Troubleshooting the Replacement Character Issue

Choosing OEM can result in an output file containing replacement characters, such as "?" or "." in place of characters that were not transformed to the OEM code page.

To fix the issue, change the encoding to the correct Unicode character set of your target file.

If some characters still fail to display correctly, do the following:

- Check that the font and script are set correctly.
- Consider using Windows Notepad or a hex viewer to view the data directly. This helps you to distinguish between transformation and display problems.

XML Connector

Set the XML connector Encoding property to UTF-8, unless otherwise specified.

Database Connectors

Setting the Correct Encoding Options During Installation

Some database and database server installations require that Unicode properties be set during initial installation. Subsequent databases that are created are sometimes limited to the Unicode or encoding properties that may not have been properly configured during the server install. For this reason, it is important to read the software manufacturer information for how to set up Unicode database server installations and subsequent databases.

ODBC Driver Support

Use an ODBC driver that supports Unicode.

Oracle Connector

See “Configuring an Oracle Database for Encoding” on page 2-8.

Sybase Connector

See “Configuring a Sybase Database for Encoding” on page 2-7.

SQL Server Connector

Set the encoding to UTF-8 and use the Unicode data types, such as nchar, nvarchar, unichar, univarchar, nlob, and nvarchar2.

CRM Connectors

When researching whether or not your CRM application supports encoding or Unicode, note that the some versions offer support

while others do not. The following CRMs support Unicode and non-Unicode encoding schemes. Note that this is not a complete list:

- Microsoft Dynamics AX
- Microsoft Dynamics CRM
- Salesforce
- RightNow CRM

Contact your software manufacturer to find out if your CRM type or version supports encoding and Unicode.

Components

Languages that use Unicode are automatically handled only if you have the Java Development Kit (JDK) installed on your system. Data Integrator installs the Java Runtime Environment (JRE), so in order to work with components you must choose a regional language setting setting options during the JDK installation.

Encoding RIFL Functions and Objects

The following RIFL functions allow you to specify encoding formats. Some of the functions include the word Unicode, however, you can set the encoding to non-Unicode encoding types as well.

Encoding Functions

- `ChangeFromUnicode`
- `UTF8Decode`
- `ChangeToUnicode`
- `B64Decode`
- `B64Encode`

When you use the following encoding File functions, you must use an encoding constant, or the function defaults to the machine's default encoding scheme, which is often ISO-8859-1. This includes nesting `File()` functions. Most often, you should use `ENC_UTF8` encoding.

File Functions

- `FileAppend`
- `FileCopy`
- `FileDateTime`
- `FileDelete`
- `FileExists`
- `FileList`
- `FileRead`
- `FileReadLine`
- `FileRename`
- `FileWrite`

The following examples show how to use the encoding parameter in File functions:

File Function Examples

```
FileRead("someFile.txt", ENC_UTF8)
FileWrite("someFile.txt", someString, ENC_UTF8)
FileAppend("someFile.txt", someString, ENC_UTF8)
```

```
FileWrite("someFile.txt", FileRead("someOtherFile.txt",  
ENC_UTF8), ENC_UTF8)
```

See Also

“Encoding Constant Values” on page A-13

RIFL Objects

The following RIFL objects allow you to change encoding parameters:

- DJMessage
- DJImport
- DJExport

DJMessage Objects

DJMessage variable properties do not support encoding. However, you can use a File() function and a message URI handle when writing or reading from a DJMessage as shown below:

```
FileRead("djmessage:///someMsg", ENC_UTF8)  
FileWrite("djmessage:///someMsg", someString, ENC_UTF8)
```

For more code examples, see the topic "Using DJMessage and DJComponent Objects in Expressions" in the *RIFL Programmer's Reference*.

See Also

“DJMessage Object Type” topic in the *RIFL Programmer's Reference Manual*

Configuring a Sybase Database for Encoding

Your server, database, and client must be properly configured to use the proper Unicode character set. You must also use data types that support Unicode data, such as unichar and univarchar. Beginning with version 12.5.1, Sybase supports the unichar and univarchar data types.

For more details on how Sybase supports encoding, refer to your Sybase documentation.

Configuring an Oracle Database for Encoding

If you have Unicode data in Oracle tables, we recommend using the ODBC 3.5 connector to connect. One exception -- if all your Unicode data has code points in the code page in use where the integration system is running, then either the ODBC 3.5 or the appropriate Oracle connector may be used.

In most cases, the Encoding property should be set to UTF-8 or UTF-16.

The default environment setting in Oracle is defined as the following:

```
Oracle NLS_LANG = language_territory.charset  
AMERICAN_AMERICA. WE8ISO8859P1  
AMERICAN_AMERICA. UTF8
```

➤ To change the locale settings in Oracle

In the Oracle database, change the language and formatting settings for your tables as needed. For instance, for the Japanese language, do the following:

```
Oracle NLS_LANG = language_territory.charset  
Japanese_Japan. JA16SJIS  
Japanese_Japan. UTF16
```

Customer Example

A company is using Data Integrator to move data from a delimited ASCII format to an Oracle database. The company is expanding its markets to include East Asian languages. The delimited text documents are mainly in UTF-8 format, and occasionally in Japanese EUC-JP and Chinese GB2312 encoding.

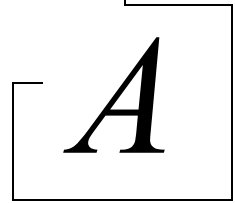
The following is the suggested approach that the company take.

- Source connector: Unicode (Delimited)
- Target connector: ODBC 3.5 (Oracle)
- Install the East Asian Language package and select **Install files for East Asian Languages**.
- The following are a few examples of the options you can set in Oracle.

Language Name	Language Type	Options
Simplified Chinese	LANG=zh_CN.GB2312	LC_ALL=zh_CN.GB2312
Traditional Chinese	LANG=zh_TW.BIG5	LC_ALL=zh_TW.BIG5
Japanese	LANG=ja_JP.eucJP	LC_ALL=ja_JP.eucJP
Korean	LANG=ko_KR.eucKR	LC_ALL=ko_KR.eucKR

For more details on how Oracle supports encoding, refer to [Oracle documentation](#).

Appendix



Supplemental Information

This appendix provides information important to selected users but is not essential to running the integration products.

- “Troubleshooting Encoding Issues” on page A-2
- “Unicode 4.0 Limitations” on page A-4
- “Unicode 4.0 Character Encodings” on page A-5
- “Customizable ICU Encodings” on page A-9
- “Encoding Constant Values” on page A-13
- “Sending Text and Binary Email” on page A-19
- “Byte Order Marks in Microsoft Windows” on page A-20

Troubleshooting Encoding Issues

When users have encoding issues while working with the data integration tools, it is usually one of the following issues:

- Not properly specifying the encoding constant when using the RIFL File() functions. See “Encoding RIFL Functions and Objects” on page 2-5.
 - Not properly specifying the encoding property when using a connector with encoding support. See “Determining Which Connector to Use” on page 2-2.
 - Using an ASCII connector instead of a Unicode connector. See “Determining Which Connector to Use” on page 2-2.
 - Not remembering to set both the source and target connector encoding properties to the correct value.
 - Not using the correct data type for encoded data. For instance, using varchar instead of the nvarchar data type.
 - Not setting the data field in the CRM or database to handle Unicode or encoded data.
 - If you cannot view your data properly, this does not mean that the data did not transform correctly from the source to the target. Ensure that the following options are set correctly: font and script settings, operating system options, Data Integrator preferences, database clients, such as TOAD, database servers, and other third-party tools, such as text viewers.
- **To test that your source data can be viewed correctly**
 - 1 Change the target connector to Unicode.
 - 2 Run the transformation transforming the source data to the target.
 - 3 View the data with a text viewer, such as Windows Notepad.
 - **To test that your target data is being output from a database correctly**

View your target data file or table in alternate viewers outside of Map Designer or Windows Notepad. For example, view the data in your database application, the CRM server, or in client views.

➤ **To determine if a file or table is Unicode**

View the data in a hex viewer to determine if there are Unicode encoding values present. Open the Structured Schema Designer to view the data with the hex browser. For instructions, search for the words “hex browser” in the online help.

See Also

“Unicode 4.0 Limitations” on page A-4

Unicode 4.0 Limitations

This implementation of Unicode has the following limitations in the data integration tool set:

Data Browser

The Map Designer data browser may not allow you to view all language combinations at the same time. If your file has characters that are not compatible with the font, script, or language settings, you may see question mark characters in the browser display. For instance, the Data Browser may display certain Vietnamese characters incorrectly. This presentation issue is not due to data loss.

➤ **To read multiple languages in your files**

Use a text editor, such as Windows Notepad or Wordpad, to view your data.

➤ **To view a single language or related language types**

Review the following settings:

- Operating System Language = [your language set] such as East Asian languages.
- Operating system fonts to read Unicode = Unicode fonts installed. For example, Arial Unicode MS is a common Windows font set.

RIFL Script Editor

The RIFL script editor does not support Unicode literal strings in scripts.

Data Parser

The Unicode (Fixed) connector is not supported in the Data Parser under these conditions:

- When UTF-8, UTF-16, or UCS-2 encoding is used.
- When the CharFieldWidths property is set to character width (True).

Unicode 4.0 Character Encodings

The integration platform supports the Unicode 4.0 character encodings listed here. The Binary (International) connector supports different encodings, listed in the topic “Binary (International) Unicode Support” in the *Source and Target Connectors User’s Guide*.

Encoding	Name
Big5	ANSI Big 5 code page
Big5 HKSCS	HK Big5
CP037	EBCDIC USA/Canada code page
CP273	EBCDIC Germany
CP277	EBCDIC Norway and Denmark
CP278	EBCDIC Sweden and Finland
CP280	EBCDIC Italy
CP284	EBCDIC Spain
CP285	EBCDIC United Kingdom
CP297	EBCDIC France
CP420	EBCDIC Arabic
CP424	EBCDIC Hebrew
CP437	DOS Latin US code page
CP500	EBCDIC International code page
CP737	DOS Greek code page
CP775	DOS Baltic Rim code page
CP838	EBCDIC Thai
CP850	DOS Latin 1 code page
CP852	DOS Latin 2 code page
CP855	DOS Cyrillic code page

Encoding	Name
CP857	DOS Turkish code page
CP860	DOS Portugese code page
CP861	DOS Icelandic code page
CP862	DOS Hebrew code page
CP863	DOS Canada code page
CP864	DOS Arabic code page
CP865	DOS Nordic code page
CP866	DOS Cyrillic Russian code page
CP869	DOS Greek 2 code page
CP870	EBCDIC Latin 2
CP871	EBCDIC Iceland
CP874	DOS Thai code page
CP875	EBCDIC Greek code page
CP918	EBCDIC Urdu
CP932	ANSI Shift-JIS code page
CP1025	EBCDIC Cyrillic
CP1026	EBCDIC Latin5 Turkish code page
CP1047	EBCDIC Open Systems Latin-1
CP1051	Roman-8
CP1097	EBCDIC Farsi
CP1112	EBCDIC Baltic
CP1122	EBCDIC Estonia
CP1123	EBCDIC Cyrillic Ukraine
CP1130	EBCDIC Vietnamese
CP1132	EBCDIC Lao
CP1250	ANSI Latin-2 code page

Encoding	Name
CP1251	ANSI Cyrillic code page
CP1252	Standard Latin-1 code page
CP1253	ANSI Greek code page
CP1254	ANSI Turkish code page
CP1255	ANSI Hebrew code page
CP1256	ANSI Arabic code page
CP1257	ANSI Baltic code page
CP1258	ANSI Vietnamese code page
EUC-CN	Extended Unix Code (EUC) encoding for Simplified Chinese
EUC-JP	Extended Unix Code (EUC) encoding for Japanese
EUC-KR	Extended Unix Code (EUC) encoding for Korean
EUC-TW	Extended Unix Code (EUC) encoding for Traditional (Taiwanese) Chinese
GB18030	ANSI Extended GB code page
GB2312	ANSI Extended GB code page
GBK	Simplified Chinese GBK
HZ	Chinese HZ
IBM950	IBM Traditional Chinese
ISO-8859-1	ISO 8859-1 standard code page
ISO-8859-2	ISO 8859-2 standard code page
ISO-8859-3	ISO 8859-3 standard code page
ISO-8859-4	ISO 8859-4 standard code page
ISO-8859-5	ISO 8859-5 standard code page
ISO-8859-6	ISO 8859-6 standard code page
ISO-8859-7	ISO 8859-7 standard code page
ISO-8859-8	ISO 8859-8 standard code page

Encoding	Name
ISO-8859-9	ISO 8859-9 standard code page
JIS7	Japanese JIS7
JIS8	Japanese JIS8
ks_c_5601	Ansi Unified Hangul code page
OEM	Corresponding OEM code page
Shift-JIS	Unicode Consortium mapping of JIS 0201 and JIS 0208
UCS-2	Standard two-byte Unicode (nonencoded)
US-ASCII	Standard US ASCII, same as ISO8859-1
UTF-8	UTF-8 encoding of Unicode (into 8-bit characters)
UTF-16	UTF-16 encoding of Unicode (into 16-bit characters)
UTF-32	UTF-32 encoding of Unicode (into 32-bit characters)
Windows-1251	Windows Cyrillic, with the Euro sign
Windows-1252	Windows Latin 1, with the Euro sign
Windows-1253	Windows Greek, with the Euro sign
Windows-1254	Windows Turkish, with the Euro sign
Windows-1255	Windows Hebrew, with the Euro sign
Windows-1256	Windows Arabic, with the Euro sign
Windows-1257	Windows Baltic, with the Euro sign
Windows-1258	Windows Vietnamese, with the Euro sign

Customizable ICU Encodings

The following table lists customizable ICU encodings.

ICU Encoding	File in Common\mappings
Big5	windows-950-2000.ucm
Big5 HKSCS	ibm-1375_P100-2003.ucm
CP037	ibm-37_P100-1995.ucm
CP273	ibm-273_P100-1995.ucm
CP277	ibm-277_P100-1995.ucm
CP278	ibm-278_P100-1995.ucm
CP280	ibm-280_P100-1995.ucm
CP284	ibm-284_P100-1995.ucm
CP285	ibm-285_P100-1995.ucm
CP297	ibm-297_P100-1995.ucm
CP420	ibm-420_X120-1999.ucm
CP424	ibm-424_P100-1995.ucm
CP437	ibm-437_P100-1995.ucm
CP500	ibm-500_P100-1995.ucm
CP737	ibm-737_P100-1997.ucm
CP775	ibm-775_P100-1996.ucm
CP838	ibm-838_P100-1995.ucm
CP850	ibm-850_P100-1995.ucm
CP852	ibm-852_P100-1995.ucm
CP855	ibm-855_P100-1995.ucm
CP857	ibm-857_P100-1995.ucm
CP860	ibm-860_P100-1995.ucm

ICU Encoding	File in Common\mappings
CP861	ibm-861_P100-1995.ucm
CP862	ibm-862_P100-1995.ucm
CP863	ibm-863_P100-1995.ucm
CP864	ibm-864_X110-1999.ucm
CP865	ibm-865_P100-1995.ucm
CP866	ibm-866_P100-1995.ucm
CP869	ibm-869_P100-1995.ucm
CP870	ibm-869_P100-1995.ucm
CP871	ibm-871_P100-1995.ucm
CP874	ibm-874_P100-1995.ucm
CP875	ibm-875_P100-1995.ucm
CP918	ibm-918_P100-1995.ucm
CP1025	ibm-1025_P100-1995.ucm
CP1026	ibm-1026_P100-1995.ucm
CP1047	ibm-1047_P100-1995.ucm
CP1051	ibm-1051_P100-1995.ucm
CP1097	ibm-1097_P100-1995.ucm
CP1112	ibm-1112_P100-1995.ucm
CP1122	ibm-1122_P100-1999.ucm
CP1123	ibm-1123_P100-1995.ucm
CP1130	ibm-1130_P100-1997.ucm
CP1131	ibm-1132_P100-1998.ucm
CP1250	ibm-5346_P100-1998.ucm
CP1251	ibm-5347_P100-1998.ucm
CP1252	ibm-5348_P100-1997.ucm
CP1253	ibm-5349_P100-1998.ucm

ICU Encoding	File in Common\mappings
CP1254	ibm-5350_P100-1998.ucm
CP1255	ibm-9447_P100-2002.ucm
CP1256	windows-1256-2000.ucm
CP1257	ibm-9449_P100-2002.ucm
CP1258	ibm-5354_P100-1998.ucm
EUC-CN	ibm-1383_P110-1999.ucm
EUC-JP	ibm-954_P101-2000.ucm
EUC-KR	ibm-970_P110-1995.ucm
EUC-TW	ibm-964_P110-1999.ucm
GB18030	gb18030.ucm
GB2312	ibm-1383_P110-1999 .ucm
GBK	windows-936-2000.ucm
IBM950	ibm-950_P110-1999.ucm
ISO-8859-2	ibm-912_P100-1995.ucm
ISO-8859-3	ibm-913_P100-2000.ucm
ISO-8859-4	ibm-914_P100-1995.ucm
ISO-8859-5	ibm-915_P100-1995.ucm
ISO-8859-6	ibm-1089_P100-1995.ucm
ISO-8859-7	ibm-813_P100-1995.ucm
ISO-8859-8	ibm-916_P100-1995.ucm
ISO-8859-9	ibm-920_P100-1995.ucm
ks_c_5601	windows-949-2000.ucm
Shift-JIS	ibm-943_P15A-2003.ucm
Windows-1251	ibm-5347_P100-1998.ucm
Windows-1252	ibm-5348_P100-1997.ucm
Windows-1253	ibm-5349_P100-1998.ucm

ICU Encoding	File in Common\mappings
Windows-1254	ibm-5350_P100-1998.ucm
Windows-1255	ibm-9447_P100-2002.ucm
Windows-1256	windows-1256-2000.ucm
Windows-1257	ibm-9449_P100-2002.ucm
Windows-1258	ibm-5354_P100-1998.ucm

**Customizing
Non-ICU
Japanese
Mappings**

For customizing any of the following Japanese encodings, please contact Technical Support Engineering.

- DEC
- IBM78EBCDIC
- IBM78EBCDIK
- IBM83EBCDIC
- IBM83EBCDIK
- JEF78EBCDIC
- JEF78EBCDIK
- JEF83EBCDIC
- JEF83EBCDIK
- JIS78
- JIS83
- KEIS78EBCDIC
- KEIS78EBCDIK
- KEIS83EBCDIC
- KEIS83EBCDIK
- MELCOM
- NECJIPSE
- NECJIPSEINT
- NECJIPSJ
- NECJIPSJINT
- UnisysLETSJ

Encoding Constant Values

The Rapid Integration Flow Language (RIFL) allows support for specific character set encodings.

Character Encoding Constants

The Encoding values listed below can be used with functions that support an **Encoding** parameter. The File functions FileRead, FileWrite, and FileAppend have optional encoding parameters, where you can specify which encoding you prefer.

The functions B64Encode, B64Decode, ChangeFromUnicode, and ChangeToUnicode also use the following encoding values.



Note If you do not specify an encoding, the default of ENC_OEM is used.

Value	Constant Name	Encoding	Description
-4	ENC_UTF16	UTF-16	UTF-16 encoding of Unicode (into 16-bit characters)
-3	ENC_UTF8	UTF-8	UTF-8 encoding of Unicode (into 8-bit characters)
-2	ENC_UCS2	UCS-2	Standard two-byte Unicode (non-encoded)
0	ENC_OEM	OEM	Corresponding OEM code page (default)
1	ENC_ISO8859_1	ISO 8859-1	Widens or narrows by ignoring the upper byte
2	ENC_CP1252	CP1252	Standard Latin-1 code page
3	ENC_ISO8859_2	ISO 8859-2	ISO 8859-2 standard code page
4	ENC_ISO8859_3	ISO 8859-3	ISO 8859-3 standard code page
5	ENC_ISO8859_4	ISO 8859-4	ISO 8859-4 standard code page
6	ENC_ISO8859_5	ISO 8859-5	ISO 8859-5 standard code page
7	ENC_ISO8859_6	ISO 8859-6	ISO 8859-6 standard code page
8	ENC_ISO8859_7	ISO 8859-7	ISO 8859-7 standard code page

Appendix

Value	Constant Name	Encoding	Description
9	ENC_ISO8859_8	ISO 8859-8	ISO 8859-8 standard code page
10	ENC_ISO8859_9	ISO 8859-9	ISO 8859-9 standard code page
11	ENC_CP1250	CP1250	ANSI Latin-2 code page
12	ENC_CP1251	CP1251	ANSI Cyrillic code page
13	ENC_CP1253	CP1253	ANSI Greek code page
14	ENC_CP1254	CP1254	ANSI Turkish code page
15	ENC_CP1255	CP1255	ANSI Hebrew code page
16	ENC_CP1256	CP1256	ANSI Arabic code page
17	ENC_CP1257	CP1257	ANSI Baltic code page
18	ENC_CP1258	CP1258	ANSI Vietnamese code page
19	ENC_SHIFTJIS	Shift-JIS	Unicode Consortium mapping of JIS 0201, JIS 0208
20	ENC_CP932	CP932	ANSI Shift-JIS code page
21	ENC_GB2312	GB2312	ANSI Extended GB code page (Simplified Chinese)
22	ENC_KSC5601	KS_C_5601	ANSI Unified Hangul code page (Korean)
23	ENC_BIG5	Big5	ANSI Big 5 code page (Traditional Chinese)
24	ENC_CP037	CP037	EBCDIC USA/Canada (English) code page
25	ENC_CP500	CP500	EBCDIC International code page
26	ENC_CP875	CP875	EBCDIC Greek code page
27	ENC_CP1026	CP1026	EBCDIC Latin 5 (Turkish) code page
28	ENC_CP437	CP437	DOS Latin US code page
29	ENC_CP737	CP737	DOS Greek code page
30	ENC_CP775	CP775	DOS Baltic Rim code page
31	ENC_CP850	CP850	DOS Latin 1 code page
32	ENC_CP852	CP852	DOS Latin 2 code page
33	ENC_CP855	CP855	DOS Cyrillic code page

Value	Constant Name	Encoding	Description
34	ENC_CP857	CP857	DOS Turkish code page
35	ENC_CP860	CP860	DOS Portuguese code page
36	ENC_CP861	CP861	DOS Icelandic code page
37	ENC_CP862	CP862	DOS Hebrew code page
38	ENC_CP863	CP863	Canada (French) code page
39	ENC_CP864	CP864	Arabic code page
40	ENC_CP865	CP865	DOS Nordic code page
41	ENC_CP866	CP866	DOS Cyrillic Russian code page
42	ENC_CP869	CP869	DOS Greek 2 code page
43	ENC_CP874	CP874	DOS Thai code page
44	ENC_CP273	CP273	EBCDIC Germany
45	ENC_CP277	CP277	EBCDIC Norway and Denmark
46	ENC_CP278	CP278	EBCDIC Sweden and Finland
47	ENC_CP280	CP280	EBCDIC Italy
48	ENC_CP284	CP284	EBCDIC Spain
49	ENC_CP285	CP285	EBCDIC United Kingdom
50	ENC_CP297	CP297	EBCDIC France
51	ENC_CP1051	CP1051	Roman-8
52	ENC_EUCJP	EUCJP	EUC-JP
53	ENC_EUCCN	EUCCN	EUC-CN
54	DJENC_EUCTW	EUCTW	EUC-TW
55	ENC_EUKR	EUCKR	EUC-KR
56	ENC_DEC	DEC	Japanese Digital VAX and Unix, DEC Kanji
57	ENC_IBM78EBCDIC	IBM78EBCDIC	Japanese IBM mainframe and AS/400, 1978 Kanji and non-kana
58	ENC_IBM78EBCDIK	IBM78EBCDIK	Japanese IBM mainframe and AS/400, 1978 Kanji and half-width kana

Appendix

Value	Constant Name	Encoding	Description
59	ENC_IBM83EBCDIC	IBM83EBCDIC	Japanese IBM mainframe and AS/400, 1983 Kanji and non-kana
60	ENC_IBM83EBCDIK	IBM83EBCDIK	Japanese IBM mainframe and AS/400, 1983 Kanji and half-width kana
61	ENC_JEF78EBCDIC	JEF78EBCDIC	Japanese Fujitsu FACOM, 1978 Kanji and non-kana
62	ENC_JEF78EBCDIK	JEF78EBCDIK	Japanese Fujitsu FACOM, 1978 Kanji and half-width kana
63	ENC_JEF83EBCDIC	JEF83EBCDIC	Japanese Fujitsu FACOM, 1983 Kanji and non-kana
64	ENC_JEF83EBCDIK	JEF83EBCDIK	Japanese Fujitsu FACOM, 1983 Kanji and half-width kana
65	ENC_NECJIPSE	NECJIPSE	Japanese NEC ACOS, NEC JIPS-E code
66	ENC_NECJIPSEINT	NECJIPSEINT	Japanese NEC ACOS, NEC JIPS-E internal code
67	ENC_NECJIPSJ	NECJIPSJ	Japanese NEC ACOS, NEC JIPS-J
68	ENC_NECJIPSJINT	NECJIPSJINT	Japanese NEC ACOS, NEC JIPS-J internal code
69	ENC_JIS83	JIS83	Japanese Public standard, Japanese Industrial Standard 1983
70	ENC_KEIS78EBCDIC	KEIS78EBCDIC	Japanese Hitachi HITACH, 1978 Kanji and non-kana
71	ENC_KEIS78EBCDIK	KEIS78EBCDIK	Japanese Hitachi HITACH, 1978 Kanji and half-width kana
72	ENC_KEIS83EBCDIC	KEIS83EBCDIC	Japanese Hitachi HITACH, 1983 Kanji and non-kana
73	ENC_KEIS83EBCDIK	KEIS83EBCDIK	Japanese Hitachi HITACH, 1983 Kanji and half-width kana
74	ENC_UnisysLETSJ	UnisysLETSJ	Japanese Unisys UNIVAC, UNIVAC LETS-J Kanji code
75	ENC_MELCOM	MELCOM	Japanese Mitsubishi MELCOM, Kanji code
76	ENC_JIS78	JIS78	Japanese Public standard, Japanese Industrial Standard 1978

Value	Constant Name	Encoding	Description
77	ENC_IBM937EBCDIC	IBM937EBCDIC	Traditional Chinese IBM mainframe, AS/400
78	ENC_IBM950	IBM950	Traditional Chinese IBM mainframe
79	ENC_GB18030	GB18030	Chinese GB18030
80	ENC_BIG5_HKSCS	BIG5_HKSCS	HK Big5
81	ENC_CP851	CP851	Simplified Chinese GBK
82	ENC_WINDOWS_1251	WINDOWS_1251	Windows Cyrillic, with the Euro sign
83	ENC_WINDOWS_1252	WINDOWS_1252	Windows Latin 1, with the Euro sign
84	ENC_WINDOWS_1253	WINDOWS_1253	Windows Greek, with the Euro sign
85	ENC_WINDOWS_1254	WINDOWS_1254	Windows Turkish, with the Euro sign
86	ENC_WINDOWS_1255	WINDOWS_1255	Windows Hebrew, with the Euro sign
87	ENC_WINDOWS_1256	WINDOWS_1256	Windows Arabic, with the Euro sign
88	ENC_WINDOWS_1257	WINDOWS_1257	Windows Baltic, with the Euro sign
89	ENC_WINDOWS_1258	WINDOWS_1258	Windows Vietnamese, with the Euro sign
90	ENC_ISO_2022_JP	ISO_2022_JP	Japanese ISO-2022
91	ENC_ISO_2022_JP_1	ISO_2022_JP_1	Japanese ISO-2022-1
92	ENC_ISO_2022_JP_2	ISO_2022_JP_2	Japanese ISO-2022-2
93	ENC_JIS7	JIS7	Japanese JIS7
94	ENC_JIS8	JIS8	Japanese JIS8
95	ENC_ISO_2022_KR	ISO_2022_KR	Korean ISO-2022
96	ENC_ISO_2022_CN	ISO_2022_CN	Chinese ISO-2022
97	ENC_ISO_2022_CN_EXT	ISO_2022_CN_EXT	Chinese ISO-2022-EXT
98	ENC_HZ	HZ	Chinese HZ
99	ENC_CP420	CP420	EBCDIC Arabic (all presentation shapes)
100	ENC_CP424	CP424	EBCDIC Hebrew
101	ENC_CP838	CP838	EBCDIC Thai
102	ENC_CP870	CP870	EBCDIC Latin 2

Appendix

Value	Constant Name	Encoding	Description
103	ENC_CP871	CP871	EBCDIC Iceland
104	ENC_CP918	CP918	EBCDIC Urdu
105	ENC_IBM930EBCDIC	IBM930EBCDIC	EBCDIC host mixed Katakana-Kanji
106	ENC_IBM933EBCDIC	IBM933EBCDIC	EBCDIC mixed Korea
107	ENC_IBM939EBCDIC	IBM939EBCDIC	EBCDIC mixed Latin-Kanji
108	ENC_CP1025	CP1025	EBCDIC Cyrillic
109	ENC_CP1047	CP1047	EBCDIC Open systems Latin 1
110	ENC_CP1097	CP1097	EBCDIC Farsi
111	ENC_CP1112	ENC_CP1112	EBCDIC Baltic
112	ENC_CP1122	CP1122	EBCDIC Estonia
113	ENC_CP1123	CP1123	EBCDIC Cyrillic Ukraine
114	ENC_CP1130	CP1130	EBCDIC Vietnames
115	ENC_CP1132	CP1132	EBCDIC Lao
116	ENC_UTF32	UTF32	UFT32
117	ENC_UTF32BE	UTF32BE	UFT32 BE
118	ENC_UTF32LE	UTF32LE	UFT32 LE
	<p>Note: UCS-2 is no longer considered a valid encoding name, however you may use UCS2. You must open the XML file with a text editor and change "UCS-2" to "UCS2".</p>		

Sending Text and Binary Email

Learn how to configure the encoding to send text and binary email in Process Designer. For details, see the IMAP, POP3, and SMTP topics in the *Component Reference*.

Byte Order Marks in Microsoft Windows

You should be familiar with how Unicode data is represented from a binary standpoint. For example, if a field is set with a size of x and x is not large enough to handle incoming double-byte data, the data may be truncated. Microsoft Windows often uses a byte order mark (BOM) with leading bytes at the beginning of a data stream when creating data files to distinguish between ASCII and Unicode data.

For instance, in a binary view of your data file, you may have extra bytes at the beginning of the file. These bytes are not necessary when handling typed data in databases and are removed during the transformation. Usually, a BOM wastes space and complicates string concatenation.

Index

C

character mappings

ICU 1-6

other than Unicode A-12

I

ICU customization 1-6

International Components for Unicode 1-6

U

Unicode custom character mappings 1-6

Unicode support A-4

Unicode table of supported encodings A-5

X

XML connector encoding property 2-3

