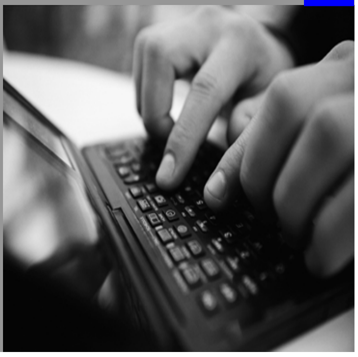
How-to Guide
SAP NetWeaver 7.0



How To... Work with Character Encodings in Process Integration

Version 1.10 – October 2007

Applicable Releases:
SAP NetWeaver 2004
Exchange Infrastructure 3.0
SAP Netweaver 7.0
Process Integration
Enabling Application-to-Application Processes

1 Scenario

You want to connect systems which have different codepages (character sets).

2 Introduction

Each XI system is based on Unicode and these messages are usually XML-based. When connecting XI with other systems it might first be necessary to convert the XI message into other codepages like Latin, Hebrew, Chinese, Japanese, Arabic and so on. However messages can also be converted from XML to plain text format, or they can be either zipped or encoded. It is therefore important to know how the transformation from the Unicode to another codepage is to be done.

Each codepage has an identifier, for example ISO-8859-1 is West European Latin, and GB18030 is a standard for Chinese characters. For each language there are several codepages available which are not always compatible, therefore you must check which codepage is used for each system.

Unicode is a standard which contains almost every character that exists in the world's languages. Unicode can be represented, among others, by the double-byte variable-length character encoding UTF-16 or the single-byte variable-length character encoding UTF-8 in which a character can consist of just 1 byte and up to 4 bytes. XI messages use UTF-8 as standard character encoding and all adapters can convert the codepage of the sender to UTF-8 as well as UTF-8 to the codepage of the receiver.

UTF-8 and all ISO-8859 codepages are based on ASCII. Therefore they are compatible when only printable characters from ASCII are used. Other characters like European ä, ê, ñ, š and letters from other alphabets are represented differently or are only available in specific codepages. These characters cause errors in interpreting XML messages if the wrong codepage is used.

Some adapters get the information about the codepage from the sender or receiver system and do the transformation automatically whilst other adapters need this information in the configuration.

The transformation between codepages is carried out by operating system routines. If you want to work with different codepages then you have to make sure that the codepages are implemented in the operating system where XI or the adapter framework is installed.

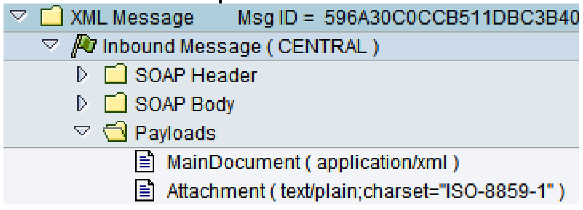
There are two places where the codepage of a message has to be declared:

- The codepage for a text message is taken from the HTTP header *Content-Type* with the attribute *charset*. For example:
  Content-Type = text/plain; charset="UTF-8"
- The codepage for an XML message is taken from the attribute "encoding" of the XML declaration. For example:
  <?xml version="1.0" encoding="UTF-8"?>

If the declaration of the codepage is missing then a default value is used which depends on the MIME type. The default for text documents is ASCII, for application/xml it is UTF-

8. There must not be a mismatch between the above-mentioned declarations in an XML message.

Here is an example taken from the XI message monitor:



```
XML Message        Msg ID = 596A30C0CCB511DBC3B40
  Inbound Message ( CENTRAL )
    SOAP Header
    SOAP Body
    Payloads
      MainDocument ( application/xml )
      Attachment ( text/plain;charset="ISO-8859-1" )
```

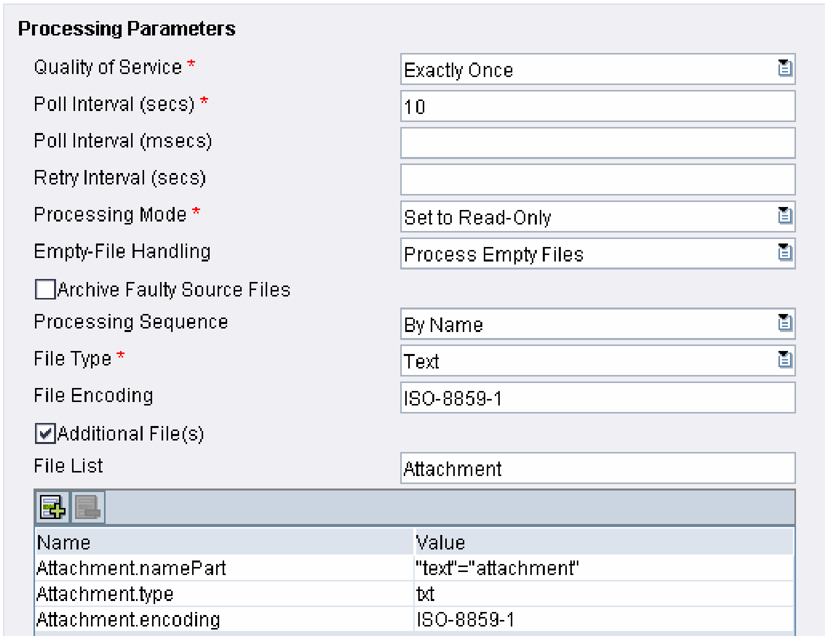
A message with two payloads is displayed. Left is the payload name and in parenthesis the content-type. The main payload with the name *MainDocument* is an XML document, the attachment with the name *Attachment* is a text document with codepage latin-1.

3 Adapter-Specific Settings

This chapter covers typical use cases. It should help you to understand how XI generally uses codepage declarations and it gives some hints if your scenario is not listed here.

3.1 File/FTP Adapter

In the sender file adapter it is necessary to apply the codepage for any text document. If you want to send several documents with the same message then you have to apply the codepage for each text document individually.
Here is an example configuration for a file sender with a configuration that enables an additional file. Both files are of type plain text and the encoding is latin-1.



Processing Parameters

Quality of Service *	Exactly Once
Poll Interval (secs) *	10
Poll Interval (msecs)	
Retry Interval (secs)	
Processing Mode *	Set to Read-Only
Empty-File Handling	Process Empty Files
☐ Archive Faulty Source Files	
Processing Sequence	By Name
File Type *	Text
File Encoding	ISO-8859-1
☑ Additional File(s)	
File List	Attachment

Name	Value
Attachment.namePart	"text"="attachment"
Attachment.type	txt
Attachment.encoding	ISO-8859-1

All text documents are converted into UTF-8. If you want to leave the text documents in their original codepage, declare the documents as binary. If you have binary files such as zip archives, declare them as binary, too.

In the receiver file adapter you can apply the codepage to which the documents are converted. This is only useful if you convert the content to a text or CSV file. If you want to change the codepage for XML you have to be aware that the codepage in the file receiver channel setting does not influence the XML declaration. Use the adapter module **`XmlAnonymizerBean`** instead (refer to chapter 4.3) or use your own XSLT or Java mapping program.

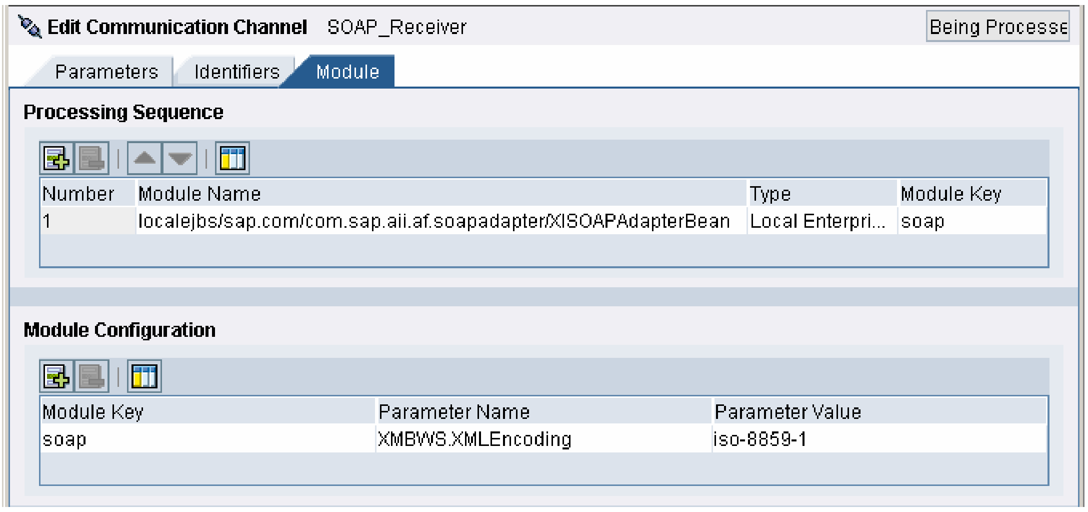
More information: SAP Note 821267: FAQ: XI 3.0 / PI 7.0 File Adapter

3.2 SOAP Adapter

The sender SOAP adapter takes the codepage information of the incoming message from the XML declaration and the content type, which must be consistent. The encoding of the response message is per default UTF-8. If you want the response sent in another codepage then you have to add the attribute **`xmlenc`** to the query string of the SOAP adapter URL, for example:

`http://host:port/XISOAPAdapter/MessageServlet?channel=p:s:c&xmlenc=iso-8859-1`

In the receiver SOAP adapter the default codepage is UTF-8. If you want to change this, add the parameter **`XMBWS.XMLEncoding`** in the module configuration for the SOAP adapter module.



Number	Module Name	Type	Module Key
1	localejbs/sap.com/com.sap.aii.af.soapadapter/XISOAPAdapterBean	Local Enterpri...	soap

Module Key	Parameter Name	Parameter Value
soap	XMBWS.XMLEncoding	iso-8859-1

More information: SAP Note 856597 - FAQ: XI 3.0 SOAP Adapter

3.3 Mail Adapter

The sender mail adapter takes the codepage information for the e-mail message from the HTTP header field content-type. In some e-mail messages the codepage is not provided, for example the content-type might be an application/octet-stream.
If codepage information is not available then you can use the module **`MessageTransformBean`** to provide the correct content-type.
More information: Chapter 4.1

In the receiver mail adapter you have the option to use the mail package. In this case you have to provide an XML structure. A description is provided in SAP Note 748024. Below is an example of a mail package (not all possible elements are used here):

```
<ns:Mail xmlns:ns="http://sap.com/xi/XI/Mail/30">
  <Subject>Hello</Subject>
  <From>sender@sender.com</From>
  <To>receiver@receiver.com</To>
  <Content_Type>text/plain;charset="UTF-8"</Content_Type>
  <Content>This is a mail</Content>
</ns:Mail>
```

Do not put the codepage to the attribute *encoding*. This attribute is used for transfer-encoding and only the values *base64* and *quoted-printable* are allowed.
More information: Chapter 6.1

When you do not use the mail package you should use the module **`MessageTransformBean`** to set the content-type. The default content-type for a mail without attachments is: application/xml.

More information: SAP Note 856599 - FAQ: XI 3.0 Mail Adapter

4 Using Adapter Modules

When you enhance the processing of the adapters by using adapter modules you have to be aware that there might be some restrictions concerning the codepage conversion, however you can also use specific modules to influence the behavior of other modules.

4.1 MessageTransformBean

Among other functions, this module enables you to set the correct content-type and codepage for a message payload which does not have this information already. This can happen for example in a mail message. Aside from this, this module also performs a codepage conversion when the codepage of the origin is clear.
The parameter for this purpose is:

```
Transfer.ContentType = <MIME type/sub type>;charset=”<charset>”
```

This module does not change the encoding declaration in the html header or xml header of the document. For XML documents use the XmlAnonymizerBean (chapter 4.3)

More Information: SAP Note 793922 - XI 3.0 AF Message Transformation Module

4.2 TextCodepageConversionBean

In SAP NetWeaver Exchange Infrastructure 3.0 SP18 and SAP NetWeaver Process Integration 7.0 SP09 a module is provided to change the codepage of a message payload. The module only works for text documents; that means the MIME type is text (for example text/plain).

The parameter is:

```
Conversion.charset = <charset>
```

This module does not change the encoding declaration in the html header or xml header of the document. For XML documents use the XmlAnonymizerBean (chapter 4.3).

More information: SAP Note 960663 - XI 3.0 AF Text Codepage Conversion Module

4.3 XmlAnonymizerBean

Starting from SAP NetWeaver Exchange Infrastructure 3.0 SP16 you can use this module to change the encoding of an XML document.
The parameter for this purpose is:

```
anonymizer.encoding = <charset>
```

You have to define all namespaces and prefixes of the XML document in the parameter **`anonymizer.acceptNamespaces`** to prevent them from being removed.

More information: SAP Note 880173 - XI 3.0 Adapter Framework XML Anonymizer Module

5 Other Techniques for Influencing the Codepage

5.1 XSLT Mapping

You can use the following XSLT Mapping to change the codepage of an XML document. In the example the encoding ISO-8859-1 is used. Change it to the required encoding of the target XML.

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" encoding="ISO-8859-1"/>
<xsl:template match="/">
<xsl:copy-of select="*" />
</xsl:template>
</xsl:stylesheet>
```

5.2 Java Mapping

You can use the following Java mapping to change the codepage of a text document. The sample program changes UTF-8 to ISO-8859-1. Change the constants at the beginning of the program according to your needs.

```
package com.sap.encoding.example;

import java.io.*;
import java.util.*;
import com.sap.aii.mapping.api.*;

public class UTFISO implements StreamTransformation {

	private final String latin = "ISO-8859-1";
	private final String utf ="UTF-8";

	public void execute(InputStream in, OutputStream out) {

		try {
			DataInputStream stdin = new DataInputStream(in);
			int length = getLengthFromStream(stdin);
			stdin.reset();
			byte[] buffer = getBytesFromStream(stdin);
			String str = new String(buffer, utf);

			str = str.replaceAll(utf, latin);
			out.write(str.getBytes(latin));
			out.close();

		} catch (IOException e) {
		}
```

```
      }

      public int getLengthFromStream(InputStream is ) throws
IOException{
            int i = 0;
            int length = 0;

            try{
                  while ((i = is.read())> 0){
                        length ++;
                  }
            }catch (ArrayIndexOutOfBoundsException e) {
                  e.printStackTrace();
            }
            return length;
      }

      public byte[] getBytesFromStream(InputStream is) throws
IOException {

                        // Create the byte array to hold the data

            byte[] bytes = new byte[getLengthFromStream(is)];
            is.reset();
                        // Read in the bytes
                        int offset = 0;
                        int tmp = 0;

                      while (offset < bytes.length
                            && (is.read(bytes, offset, offset +
(tmp = is.available()))) > 0) {
                          offset += tmp;
                      }

                      // Ensure all the bytes have been read in
                      if (offset < bytes.length) {
                             throw new IOException("Could not
completely read Inputstream");
                      }
                      // Close the input stream and return bytes
                      is.close();
                      return bytes;
             }

      public void setParameter(Map param) {
      }

}
```

6 Related Issues

Besides choosing a codepage for representing a text document correctly, there are other mechanisms for the encoding of a file. These techniques are independent from the codepage and must not be confused.

6.1 Content-Transfer Encoding

Content-transfer encoding is a technique for representing files by ASCII characters. This should help to transfer these files between different servers. Two standards are usually used:

- Quoted-Printable:
  This is mainly used for non-ASCII text files where all non-ASCII characters are replaced by an escape sequence.
- Base64:
  This is mainly used for binary files where the whole file is represented by a sequence of ASCII characters.

The XI sender adapters decode automatically when the appropriate attribute in the HTTP header is set.

6.2 Escape Sequences

The escape sequences are necessary to distinguish XML content and control signs such as &, < and >. The escape sequences start with an ampersand (&). For example: ‘&amp;’ represents ‘&’ and ‘&gt;’ represents ‘>’.
All XI components that convert between text and XML perform the escaping and de-escaping of above-mentioned characters. In some cases, an external sender of an XML message does not provide this functionality and therefore a parsing error occurs whenever an ampersand appears in the XML document. The only solution is to use a Java mapping before the actual mapping to perform the escaping.