Configuring Character Encoding for Input Documents in Data Transformation
Abstract
Each type of input document has its own character encoding. This article describes the supported encodings and how to configure the encoding for each input document.

Supported Versions
- Data Transformation 9.1.0

Table of Contents
- Overview .......................................................... 2
- Document Encoding .............................................. 3
  - Supported Code Pages ........................................... 3
- Configuring Input Encoding for the Main Input ................. 4
- Configuring Input Encoding for an Additional Input Port ........ 5

Overview
A character encoding defines the number of bytes for each character and the mapping between the characters and their binary equivalents. The input settings must match the encoding for each input document.

The input encoding in the project properties applies to the main input, when it is not configured with an AdditionalInputPort. The main input processes the following documents:
- The sample document when you run the project in the Studio.
- The document specified through an API when you run the project in the production environment.
- The set of input documents specified under Parser > sources_to_extract.

Additional input documents are passed through an AdditionalInputPort to parsers, mappers, or serializers. Configure the encoding for each additional document individually in the associated AdditionalInputPort.

Note: When an input document is processed by a parser or serializer, it is also affected by transformers that apply to all content anchors and by document processors.
The document encoding defines the number of bytes per character and the mapping between characters and their binary equivalents.

The default input encoding in Data Transformation is Windows-1252 (Latin). The following table describes the ways to change the document encoding:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project &gt; Properties &gt; Encoding &gt; Input</td>
<td>Defines the input encoding for the documents processed by the main parser, mapper, or serializer. For more information, see &quot;Configuring Input Encoding for the Main Input&quot; on page 4.</td>
</tr>
<tr>
<td>AdditionalInputPort &gt; input_encoding &gt; code_page</td>
<td>Defines the input encoding for the document processed through an AdditionalInputPort. For more information, see &quot;Configuring Input Encoding for an Additional Input Port&quot; on page 5.</td>
</tr>
</tbody>
</table>

You can use the Encoder transformer to change any encoding. In the project properties, you can also change the working encoding and the output encoding.

**Supported Code Pages**

Data Transformation supports the following code pages:

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Big5</td>
<td>Chinese</td>
</tr>
<tr>
<td>Big5-HKSCS</td>
<td>Chinese with Hong Kong Supplementary Character Set</td>
</tr>
<tr>
<td>EBCDIC-37</td>
<td>US/Canada</td>
</tr>
<tr>
<td>EBCDIC-284</td>
<td>Spanish</td>
</tr>
<tr>
<td>EBCDIC-424</td>
<td>Hebrew</td>
</tr>
<tr>
<td>EUC-KR</td>
<td>Korean</td>
</tr>
<tr>
<td>GB2312</td>
<td>Chinese</td>
</tr>
<tr>
<td>GB18030</td>
<td>Chinese</td>
</tr>
<tr>
<td>ISO-8859-1</td>
<td>Latin-1 (English and West European)</td>
</tr>
<tr>
<td>ISO-8859-2</td>
<td>Latin-2 (East European)</td>
</tr>
<tr>
<td>ISO-8859-3</td>
<td>Latin-3 (South European)</td>
</tr>
<tr>
<td>ISO-8859-4</td>
<td>Latin-4 (North European)</td>
</tr>
<tr>
<td>ISO-8859-5</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>ISO-8859-6</td>
<td>Arabic</td>
</tr>
<tr>
<td>Encoding</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>ISO-8859-7</td>
<td>Greek</td>
</tr>
<tr>
<td>ISO-8859-8</td>
<td>Hebrew</td>
</tr>
<tr>
<td>ISO-8859-9</td>
<td>Latin-5 (Turkish)</td>
</tr>
<tr>
<td>ISO-8859-15</td>
<td>Latin-9 (West European)</td>
</tr>
<tr>
<td>KSC_5601</td>
<td>Korean</td>
</tr>
<tr>
<td>Shift_JIS</td>
<td>Japanese</td>
</tr>
<tr>
<td>TIS-620</td>
<td>Thai</td>
</tr>
<tr>
<td>UTF-16</td>
<td>Unicode</td>
</tr>
<tr>
<td>UTF-16BE</td>
<td>Unicode (big-endian)</td>
</tr>
<tr>
<td>UTF-7</td>
<td>Unicode</td>
</tr>
<tr>
<td>UTF-8</td>
<td>Unicode</td>
</tr>
<tr>
<td>Windows-874</td>
<td>Thai</td>
</tr>
<tr>
<td>Windows-1250</td>
<td>Central European</td>
</tr>
<tr>
<td>Windows-1251</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>Windows-1252</td>
<td>ANSI English and West European</td>
</tr>
<tr>
<td>Windows-1253</td>
<td>Greek</td>
</tr>
<tr>
<td>Windows-1254</td>
<td>Turkish</td>
</tr>
<tr>
<td>Windows-1255</td>
<td>Hebrew</td>
</tr>
<tr>
<td>Windows-1256</td>
<td>Arabic, Farsi, Urdu</td>
</tr>
<tr>
<td>Windows-1257</td>
<td>Baltic</td>
</tr>
<tr>
<td>Windows-1258</td>
<td>Vietnamese</td>
</tr>
</tbody>
</table>

**Configuring Input Encoding for the Main Input**

The input encoding defined in the project properties defines the encoding of the documents that are passed through the main input.

1. Click **Project > Properties**.  
   The Properties for [ProjectName] window appears.
2. In the left panel, select Encoding.
3. In the Input panel, click Custom, and then select an encoding from the list.
4. Click OK and then save the project.
Configuring Input Encoding for an Additional Input Port

When a parser, mapper, or serializer reads a document in through an AdditionalInputPort, it interprets the document according to the encoding defined for the AdditionalInputPort.

1. To the right of the AdditionalInputPort > input_encoding property, double-click the ellipsis (...), and then select PortEncoding.
2. To the right of the input_encoding > code_page property, double-click the ellipsis (...), and then select an encoding from the list.
3. Select one of the following options for the encode_as_xml property:
   - Selected. XML control symbols are represented as XML entities, for example, &lt;
   - Cleared. XML control symbols are represented as ASCII characters, for example, <

Author

Charles Dorsett
Senior Technical Writer

Acknowledgements

The author would like to acknowledge the contributions of Etai Margolin and Or Arieli.