Pentaho Internationalization

Recommendations For Fixing/Improving Ability To Internationalize The Pentaho Platform

Encoding History

US ASCII

- 7 bit encoding supports 127 english characters (code points) and punctuations.

ISO-8859

- 8 bit encoding supports 255 code points.
- First 127 code points are same as US ASCII
- Code points 128 – 159 not part of standard due to lack of consensus.
- Code points 160 – 255 are used over and over with different meaning (Latin 1, Latin 2, etc)
- English windows uses Latin 1 variant (Cp1252) ,with custom characters defined for code points 128 – 159.

Unicode

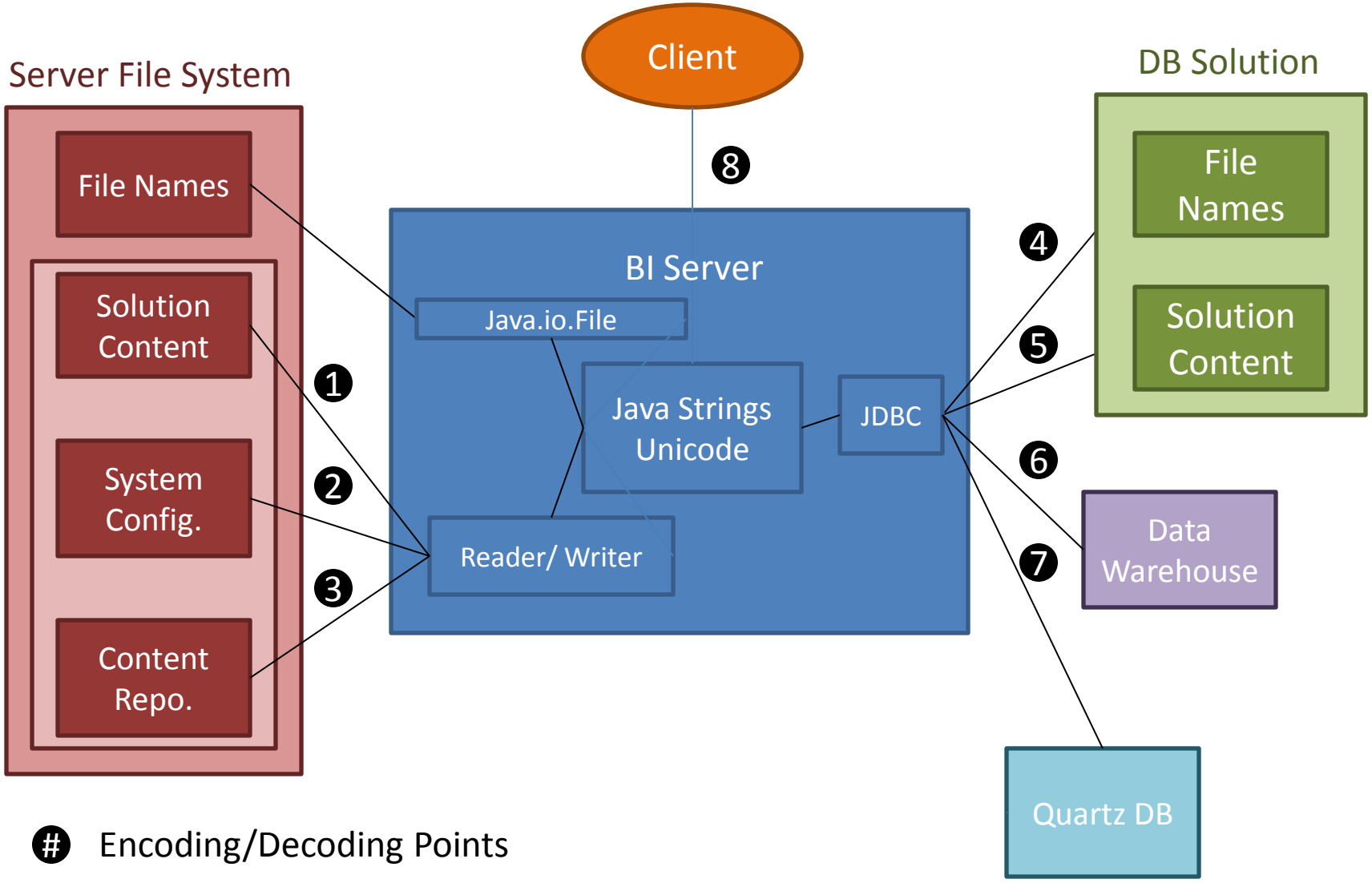
- 16 bit encoding
- Millions of available code points About 100,000 are used.
- First 127 code points are the same as US ASCII.

UTF -8

- Variable length encoding (1 – 4 bytes per character).
- Supports all currently defined unicode characters.
- First 127 code points same as US ASCII.
- Code points 160 – 255 used to determine # of following bytes pertain to character.

Many Other Encoding Available

Pentaho Platform Encoding



Encoding/Decoding Points

Encoding Recommendations

Customers Using One Single-Byte Encoding (i.e. Latin-1 or Latin-2 or SHIFT_JIS, etc.)

- Server File System
  - Server System Encoding defined at user's discretion
  - Solution/config file content should use default system encoding . This is preferable because admins can easily edit the file using any text editor.
  - Content repo is UTF-8 encoded regardless of system encoding.
- DB Solution/Quartz
  - DB encoding needs to support character set (same encoding or UTF-8).
- Data Warehouse
  - Encoding may differ from above but, string content must be compatible with the encoding chosen above.

Multinational Customers

- Server File System
  - Recommended File Server System Encoding is UTF-8
  - Solution/config file content should use default system encoding (i.e. UTF-8).
  - Can override default encoding using JVM parameter which affects all webapps running on the VM or using a Pentaho paramter. Which only affects the Pentaho webapp.
  - Content repo is UTF-8 encoded regardless of system encoding.
- DB Solution/Quartz
  - DB needs to support UTF-8
- Data Warehouse
  - Unknown, and don't care because UTF-8 above will handle all characters

Coding Recommendations

- Get rid of LocaleHelper.getSystemEncoding().
- Use Readers and Writers to process character data from streams.
- Decode character data (convert to java.lang.String) as soon as possible.
- Encode character data (remain as java.lang.String) as late as possible.
- In the case of the solution repository all encoding/decoding should be happening int the file based and db based solution repository classes.
- Don't pass around known character data as byte arrays as this requires the encoding to be passed with the array.
- When writing XML using an editor:
  - Always use the encoding attribute
  - Use an editor that supports encoding
  - Make sure you know what encoding the editor uses
  - Use the same encoding in your encoding attribute
- When writing XML programmatically:
  - Always use the encoding attribute
  - Use a Writer and specify the same encoding in your encoding attribute