IBM z Systems

TPFUG – JavaScript Object Notation (JSON)

Colette A. Manoni, TPF Development Lab

March 23, 2015

IBM

Disclaimer

- Any reference to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.

JavaScript Object Notation (JSON)

- Is a lightweight data-interchange format
- Based on a subset of JavaScript programming language
  - Text format makes it completely language independent
- Uses 2 constructs / data structures
  - Name : value pairs
  - Ordered list of values

```
{
  "anObject": {
    "aNumber": 147,
    "aString": "Hello world!",
    "aBoolean": true,
    "aDate": "2015-03-23"  },

  "arrayOfObjects": [
    { "item": 1  },
    { "item": 2  },
    { "item": 3  }  ]
}
```

Why JSON?

- Document is smaller in size than XML
- Parsers available in all different languages
- Can be used easily from JavaScript

- REST/JSON popular replacement to SOAP/XML
- Widely used in mobile applications

- Interoperability with TPF
  - Can pass-thru data from mobile applications
  - Streamline the processing - don’t need a converter in the middle

JSON Parser/Generator on z/TPF

- PJ42279 – available with PUT11

- Design points
  - Standards
    - Investigated porting an existing parser
    - No standard API exists for JSON
  - Performance
    - Key is to keep processing lightweight
    - Basic measurements on-par or better than distributed parsers
    - Better performance (~5%) than XML parser
  - Integration with existing z/TPF functions
    - Infonodes
    - DFDL

The APIs: tpf_doc_*

- APIs structured after tpf_xml_*
  - tpf_doc_* APIs can be used to parse and generate either XML or JSON documents

- New capabilities with tpf_doc_*
  - tpf_doc_getElement and tpf_doc_addElement
    - Elements accessed using a fully qualified path notation starting from the root
    - Child nodes are specified through the use of the '.' character.
    - Array nodes (only available with JSON tree structures) are specified using the '[' and ']' surrounding the desired array index. Multi-dimensional arrays are supported.
    - The quote character '"' may be used to surround node specification to include other special characters as part of the the node name.

Example: tpf_doc_addElement and tpf_doc_getElement

JSON document:

{ a : 1, b : [1, 2, {c: 1}] }

```
tpf_doc_addElement("a","1");
tpf_doc_addElement("b[0]","1");
tpf_doc_addElement("b[1]","2");
tpf_doc_addElement("b[2].c","1");
```

```
tpf_doc_getElement("a")  = "1"
tpf_doc_getElement("b[0]") = "1"
tpf_doc_getElement("b[1]") = "2"
tpf_doc_getElement("b[2].c") = "1"
```

Data type: TYPE_RAW_TEXT

- JSON documents are usually encoded using Unicode (UTF-8)
- With the parser, all data is converted to EBCDIC to enable application processing
  - Not all UTF-8 characters can be converted to EBCDIC
  - May sometimes get the substitution character – corrupts original data
- tpf_doc_* APIs provide option to preserve the original encoding
  - Always preserved when parsing JSON
  - Option to preserve when parsing XML
- Can access the original string via the xmlNodesArray structure by specifying TYPE_RAW_TEXT for the data type

```
xmlNodesArray *tpf_doc_getElement (XMLHandle api_handle,
                                   char * elementPath,
                                   XML_DATA_TYPE data_type)

xmlNodesArray->nodesArray[x].nodeValueStr   will contain original Unicode value
```

Questions?

Thank you!

Trademarks

- IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Notes

- Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.
- All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.
- This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.
- All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.
- Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.
- Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.
- This presentation and the claims outlined in it were reviewed for compliance with US law. Adaptations of these claims for use in other geographies must be reviewed by the local country counsel for compliance with local laws.