z/OS 3.1 IBM Education Assistant

Solution Name: JSON parser comment toleration

Solution Element(s): z/OS client web enablement toolkit (JSON parser)

July 2023

Agenda

- Trademarks
- Objectives
- Overview
- Usage & Invocation
- Interactions & Dependencies
- Upgrade & Coexistence Considerations
- Installation & Configuration
- Summary
- Appendix

Trademarks

- See http://www.ibm.com/legal/copytrade.shtml for a list of trademarks.
- Additional Trademarks:
  - None

Objectives

Describe the JSON comment toleration enhancement for the JSON parser portion of the z/OS client web enablement toolkit.

Overview

- Who (Audience)
  - Users of the z/OS JSON parser

- What (Solution)
  - JSON parser added support to tolerate single and multi-line comments inside JSON data, per JSON5 specifications.

- Wow (Benefit / Value, Need Addressed)
  - JSON content, while human-readable, can become **overwhelming to read** on its own.
  - Customers may want to provide information on **how to maintain JSON** configuration files within those files themselves.
  - This enhancement allows customers to **solve both readability and maintainability** problems by adding comments to their JSON data.

Usage & Invocation

The z/OS JSON parser will successfully parse JSON text that contains single and multi-line comments as defined by the JSON5 Data Interchange Format extension to JSON. All other syntax parsing will continue to follow the base JSON specification.

NOTE: This comment support is **toleration only**; the application will not be able to retrieve any of the comments detected in the JSON text and the contents produced by HWTJSERI API invocation will exclude any comments that may have originally been present.

```
{
"policyName":     "         JobConverALLJC01         ",
"policyVersion":  1,
"policyType":     " JobConversion ",
/*
    Policy contains all attributes, actions as of 2.5 PID for
    JobConversion.

    Attributes:  (24)
            CompletionCode, CreatedLocally, InputMember, JobAcct,
            JobClass, JobHasAffinity( member-list), JobHasFailed,
            JobIsHeld, JobIsDupl_Exempt, JobIsProtected,
            JobIsPrivileged, JobName, JobOwner, JobPgmList, JobPrty,
            JobSubmitter, JobSecLabel, JobType, MsgClass, SysAff,
            SrvClass, SchEnv, UJobCorr, WithJob

            MODIFIABLE:
                JobClass, JobIsDupl_Exempt, JobPrty, SysAff,
                SrvClass, SchEnv, WithJob

    Actions:
            ModifyJob, HoldJob, CancelJob,
            Leave, SendMessage, LogMessage


            ***** Testing All  ***************************
            * All attributes are tested inside their       *
            * condition and displayed with send/log        *
            * message. All modifiable attributes are then  *
            * modified within their own condition def      *
            * then standard actions                        *
            ***********************************************

 */
"definitions":
  [
    { /******** Testing CompletionCode ***************/
      "condition" : " String(CompletionCode) = '()' ",
      "actions" :
      [
        {"action" : "sendMessage",
         "message": " 'CompletionCode: '||String(CompletionCode) "}
      ]
```

Usage & Invocation – HWTJOPTS API

Users can modify parser behavior by taking advantage of the new **HWTJOPTS API**

- This service, if invoked, must be invoked prior to the first successful "parse", either using HWTJPARS, or HWTJCREN if authoring new JSON text.

- Supported values are:
  - **HWTJ_TOLERATE_CMT_OFF**
    The z/OS JSON parser should comply with the original JSON specification and classify any JSON5 single and multi-line style comments encountered as invalid syntax.

  - **HWTJ_TOLERATE_CMT_ON**
    The z/OS JSON parser should be primed to expect single and multi-line comments during parse, HWTJPARS API, and creation, HWTJCREN API, of JSON text. Parsing performance for commented JSON text is slower compared to uncommented JSON. This option is only recommended for content that will contain comments.

Usage & Invocation – Default behavior

The **default parser behavior** will be to tolerate comments with preferential treatment for uncommented JSON text.

By default the parser will continue to deliver **optimum performance for uncommented JSON** text. The parsing performance for **commented JSON text will always be slower** in comparison but can be optimized by including a comment as early in data as possible.

- **Commented JSON Example 1** will have the best performance because the parser will detect it should tolerate comments when it reads the very first line. This is on par with performance when using **HWTJ_TOLERATE_CMT_ON**.

- **Commented JSON Example 3** will have the worst performance because the parser will not be aware it needs to tolerate comments until the very end of the text and will need to re-parse the whole text body, resulting in two full parses

Commented JSON Example 1

```
line 1 /* This JSON contains a policy definition */ <- first comment
line 2 {
line 3         "action" : "sendMessage" //first action
line 4        ,"message": " 'CompletionCode: '||String(CompletionCode)”
line 5 }
```

Commented JSON Example 2

```
line 1 {
line 2         "action" : "sendMessage" //first action <- first comment
line 3        ,"message": " 'CompletionCode: '||String(CompletionCode)"
line 4 }
```

Commented JSON Example 3

```
line 1 {
line 2         "action" : "sendMessage"
line 3        ,"message": " 'CompletionCode: '||String(CompletionCode)"
line 4 } /* The above was an action */   <- first comment
```

Usage & Invocation – Performance note

In addition to the comment toleration support, the CPU and elapsed time associated with the parsing of JSON content is **reduced by up to 50%**[1]

[1] **Disclaimer:** This reduction is based on internal measurements done on an IBM z15 using a z/OS V2.5 LPAR with 8 CPs. The z/OS JSON Parser was used to parse a 568 MB JSON input file containing public property tax records and geospatial data. The input file included 7,875,189 numbers, 3,038,859 arrays, 2,217,825 strings, 54,336 nulls, no booleans, and no comments. The maximum nesting depth of any member or element was 7 levels. Reported results were derived from measurements that tested 100 parses of the input file back to back. Results may vary.

Interactions & Dependencies

- Software Dependencies
  - None
- Hardware Dependencies
  - None
- Exploiters
  - None

Upgrade & Coexistence Considerations

- To exploit this solution, all systems in the Plex must be at the new z/OS level: No

Installation & Configuration

- Included in z/OS 3.1 base

- APAR OA61974
  - Rolled down to z/OS 2.5
  - IPL is required after applying the APAR

Summary

- Applications that invoke the z/OS JSON parser can now successfully parse commented JSON text.

Appendix

- Publications
  - z/OS MVS Programming: Callable Services for High-Level Languages

  https://www.ibm.com/docs/en/zos/2.5.0?topic=toolkit-zos-json-parser

- External references
  - https://spec.json5.org Standard JSON5 1.0.0 (Section 7)
  - https://www.json.org JSON (JavaScript Object Notation)