z/OS 2.4 IBM Education Assistance (IEA)

Solution (Epic) Name: Miscellaneous function for v2.4

Element(s)/Component(s): z/OS client web enablement toolkit

Material prepared for IBM Education Assistance

© 2019 IBM Corporation

Agenda

© 2019 IBM Corporation

Trademarks

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

Session Objectives

Describe the following z/OS client web enablement toolkit (toolkit) enhancements that are included in the z/OS 2.4 release and were also rolled back to z/OS 2.2 and z/OS 2.3.

- HTTP Proxy Enhancements
- JSON Delete API
- JSON UTF-8
- JSON Shallow Search
- Pretty-print – json formatter

© 2019 IBM Corporation

Overview - HTTP Proxy Enhancements

- Who (Audience)
  - Toolkit HTTP/HTTPS enabler users that need access to web servers via an HTTP proxy…"
    - Sub-case 1: "...**and** their proxy is an 'authenticating' proxy."
    - Sub-case 2: "...**and** their administrators require AT-TLS to provide all HTTPS security."

- What (Solution)
  - Authenticating proxy:
    - Toolkit added two new options that allow the user to specify the user name and password required by their authenticating HTTP proxy, along with another option to enable their use.

  - AT-TLS requirement:
    - Toolkit recognizes a proxy connection that allows AT-TLS to secure an end-to-end tunnel **through** (via) a proxy to an HTTPS server, and the toolkit initiates that security with AT-TLS if toolkit options allow.

- Wow (Benefit / Value, Need Addressed)
  - Programmers can now incorporate the toolkit into their client applications, even
    - when their client must use an authenticating proxy, and/or
    - when their client requires AT-TLS security for their HTTPS connection.

Usage & Invocation - HTTP Proxy Enhancements

Anytime a proxy is used, the following *existing* options **must** be specified to set the proxy address and port:

- **HWTH_OPT_PROXY**
- **HWTH_OPT_PROXYPORT**

If the proxy is an “authenticating proxy” then the application is now able to provide “Basic authentication” credentials for the proxy using the following new options, supported during connection handle initialization:

HWTH_OPT_PROXYAUTH

HWTH_PROXYAUTH_NONE

No proxy authorization is used. Default.

HWTH_PROXYAUTH_BASIC

Use basic proxy authentication. The user name and password, as specify by the **HWTH_OPT_PROXYAUTH_USERNAME** and **HWTH_OPT_PROXYAUTH_PASSWORD** options, are processed and sent to the proxy as prescribed by the Basic authentication format. See RFC 7617.

HWTH_OPT_PROXYAUTH_USERNAME

An optional buffer that contains the user name to be used as part of authorizing to a proxy that requires Basic authentication.

HWTH_OPT_PROXYAUTH_PASSWORD

An optional buffer that contains the password to be used as part of authorizing to a proxy that requires Basic authentication.

© 2019 IBM Corporation

Usage & Invocation - HTTP Proxy Enhancements

In addition, the application can now use AT-TLS to secure an HTTPS destination connection via a proxy

- AT-TLS must map the **proxy connection address** (defined by HWTH_OPT_PROXY and HWTH_OPT_PROXYPORT) to an “application-controlling” AT-TLS policy.

- The “application-controlling” policy for the HTTP proxy connection MUST specify a keyring that is suitable for use with **any and all** HTTPS destinations that will be reached via that proxy.

- Do NOT map the HTTPS destination address (defined by HWTH_OPT_URI and HWTH_OPT_PORT) to a policy unless you intend to connect to it ‘directly’ (without a proxy).

© 2019 IBM Corporation

Usage & Invocation - HTTP Proxy Enhancements

Example 1, snippet of C code that indicates proxy username "user1" and proxy password "pass1" should be used to authenticate if the connection encounters an authenticating proxy:

```
char *lcharOptionVal;
uint32_t intOption;
uint32_t *intOptionPtr = &intOption;
HWTH_SET_OPTION_TYPE loption;

loption = HWTH_OPT_PROXYAUTH;
intOption = HWTH_PROXYAUTH_BASIC;
hwthset(rcPtr, *connectionHandlePtr, loption,
       (void **) &intOptionPtr,
       sizeof(intOption), diagAreaPtr);
ASSESS(validateSuccess(rcPtr, diagAreaPtr,"HWTHSET for connection - proxy auth"));


loption = HWTH_OPT_PROXYAUTH_USERNAME;
lcharOptionVal = "user1";
hwthset(rcPtr, *connectionHandlePtr, loption,
       (void**) &lcharOptionVal,
       strlen(lcharOptionVal), diagAreaPtr);
ASSESS(validateSuccess(rcPtr, diagAreaPtr,"HWTHSET for connection - proxy username"));


loption = HWTH_OPT_PROXYAUTH_PASSWORD;
lcharOptionVal = "pass1";
hwthset(rcPtr, *connectionHandlePtr, loption,
       (void**) &lcharOptionVal,
       strlen(lcharOptionVal), diagAreaPtr);
ASSESS(validateSuccess(rcPtr, diagAreaPtr,"HWTHSET for connection - proxy password"));
```

© 2019 IBM Corporation

Usage & Invocation - HTTP Proxy Enhancements

- New reason code returned when an HTTPS connection attempts to use an authenticating proxy without valid PROXYAUTH credentials.

Sample HTTPS failure:

```
    Request failed (HWTHCONN).
    Return code: HWTH_COMMUNICATION_ERROR (‘106’x)
    Reason Code: HWTH_RSN_PROXY_AUTH_REQD (‘17’x)
```

© 2019 IBM Corporation

Usage & Invocation - HTTP Proxy Enhancements

Toolkit tracing is enhanced in correlation with this support:

- Unconditional trace of detected AT-TLS status
- New trace points that clarify which of several possible paths was followed

Toolkit tracing may be enabled via the existing HWTH_OPT_VERBOSE/ HWTH_OPT_VERBOSE_OUTPUT options set by the application on the connection handle.

```
t-Entry: iconnImpl
t-Entry: initTranslationTables
t-Exit: initTranslationTables
t: Connecting to uhclem.rtp.raleigh.ibm.com via port 443
t: Attempting to connect to IP address: 9.42.21.114
t-Entry: setSocketOptions
t-Exit: setSocketOptions
t: Connection established using socket: 1
t-Entry: checkForTTLS
t: ATTLS detection OK: s=1 pol=APPLCNTRL conn=NOTSECURE
t: Socket maps to TTLSRule: UhClem_ApplCntrl_443
t: Tolerating APPLCNTRL policy on non-proxy
t-Entry: initSSLEnv
t: Creating a new SSL environment
t: Using the default SSL protocols
t: Setting SSL key database to: *AUTH*/*

.
.
.

t: now ignoring signal: SIGPIPE
t-Exit: ignoreSignal
t: Invoke gsk_secure_socket_init()
t-Entry: restoreSignal
t: restoring signal: SIGPIPE
t-Exit: restoreSignal
t: A full handshake was required.
t-Exit: initSSLEnv
t: Socket was secured with toolkit options: 1
t-Exit: iconnImpl
t-Entry: sendrqst

-------------------------------------------------------------------

t-Entry: setupRedirect
t: Using an absolute redirect.
t: Redirect (->HTTPS)
t: Redirect URI specifies port: 51012
t: Create a short-lived connection for xProtocol redirect.
t-Entry: iconnImpl
t-Entry: initTranslationTables
t-Exit: initTranslationTables
t: Connecting to godzilla.rtp.raleigh.ibm.com via port 51012
t: Attempting to connect to IP address: 9.37.247.41
t-Entry: setSocketOptions
t-Exit: setSocketOptions
t: Connection established using socket: 1
t-Entry: checkForTTLS
t: ATTLS detection OK: s=1 pol=ENABLED conn=SECURE prot=TLSV1_2 ciph=F0F0F3F5 fip=FIPS140_OFF
t: Socket maps to TTLSRule: IBMUSER_Auto_TTLS_v12_51012
t: ATTLS detected, rule name: IBMUSER_Auto_TTLS_v12_51012, protocol version: 4
t: Connection secured by ATTLS on socket: 1
t-Exit: iconnImpl
```

Installation - HTTP Proxy Enhancements

- Rolled down to z/OS 2.2 and z/OS 2.3 via OA54902
- IPL is required after applying the APAR

© 2019 IBM Corporation

Overview – JSON Delete API

- Who (Audience)
  - Toolkit z/OS JSON parser users
- What (Solution)
  - New service that can delete content from a JSON data object
- Wow (Benefit / Value, Need Addressed)
  - This new service simplifies the modification of an existing JSON data object. The user can use this new service to delete existing content in a JSON object instead of *manually* manipulating the object.

Usage & Invocation - JSON Delete API

The new HWTJDEL service can be used to delete either an object or an array entry.

Non-REXX syntax:

```
CALL HWTJDEL(
        ReturnCode,
        ParserHandle,
        ObjectHandle,
        EntryValueHandle,
        DiagArea);
```

REXX syntax:

```
address hwjson “hwtjdel”,
        “ReturnCode”,
        “ParserHandle”,
        “ObjectHandle”,
        “EntryValueHandle”,
        “DiagArea.”
```

Usage & Invocation - JSON Delete API

Example 1: Deleting an entry from an object

Given the following JSON text:

```
{
   "foo":{
     "mood":"happy",
     "color":"red",
     "bling":"baz"
       },
  "bar":[ "bag", 3, true]
}
```

Resulting JSON text:

```
{
   "foo":{
     "color":"red",
     "bling":"baz"
       },
  "bar":[ "bag", 3, true]
}
```

To delete the “mood” entry from the “foo” object

```
/* Obtain the “foo” target object handle*/
objectName = "foo”
rootObject = 0      /* search the root object */
startingHandle = 0  /* start with the first member */

HWTJSRCH(returnCode, parserHandle,
     HWTJ_SEARCHTYPE_GLOBAL, objectName,
     length(objectName), rootObject,
     startingHandle, targetObjectHandle,
     diagArea)

/* Obtain the “mood” target entry handle */
entryName = "mood”

HWTJSRCH(returnCode, parserHandle,
     HWTJ_SEARCHTYPE_OBJECT, entryName,
     length(entryName), targetObjectHandle,
     startingHandle, targetEntryHandle,
     diagArea)

/* Delete “mood” entry from the “foo” object */
HWTJDEL(returnCode, parserHandle,
     targetObjectHandle, targetEntryHandle,
     diagArea)
```

Usage & Invocation - JSON Delete API

Example 2: Deleting an entry from an array

Given the following JSON text:

```
{
  "bar":[
          "bag",
          3,
          true,
          {
            "a": "somewhere"
          }
        ],
   "blind" : "blam",
   "pi": 3.14159
}
```

To delete the fourth entry, “a”, from the “bar” array

```
/* Obtain a handle to the "bar" target*/
arrayName = "bar"
rootObject = 0     /* Search the root */
startingHandle = 0 /* Start at the first member */
targetArrayHandle = 0

HWTJSRCH(returnCode, parserHandle,
     HWTJ_SEARCHTYPE_GLOBAL, arrayName,
     length(arrayName), rootObject,
     startingHandle, targetArrayHandle,
     diagArea)

/* Obtain a handle to the fourth array value */
/* Note that array entries are zero-indexed */
entryIndex = 3

HWTJGAEN(returnCode, parserHandle,
     targetArrayHandle, entryIndex,
     targetEntryHandle, diagArea)

/* Delete the fourth entry from the "bar" array */
HWTJDEL(returnCode, parserHandle,
     targetArrayHandle, targetEntryHandle,
     diagArea)
```

Resulting JSON text:

```
{
  "bar":[
          "bag",
          3,
          true
        ],
   "blind" : "blam",
   "pi": 3.14159
}
```

Usage & Invocation - JSON Delete API

New samples shipped in SYS1.SAMPLIB that take advantage of delete service:

- HWTJXRX2 (Rexx)
- HWTJXC2 (LE C)
- HWTJXCB2 (Cobol)

Installation - JSON Delete API

- Rolled down to z/OS 2.2 and z/OS 2.3 via OA54901
- IPL is required after applying the APAR

Overview – JSON UTF-8

- Who (Audience)
  - Toolkit z/OS JSON parser users
- What (Solution)
  - The JSON parser now supports JSON text encoded either in UTF-8 or in EBCDIC-1047
- Wow (Benefit / Value, Need Addressed)
  - This support simplifies the usage of UTF-8 data via the JSON parser. Prior to UTF-8 support, the current implementation of the JSON parser required a customer to go through the overhead of manually converting data into IBM-1047 prior to using the parser. Depending on the data, this conversion could require extraordinary effort, since IBM-1047 is limited to the ISO 8859-1 (Latin-1) subset of Unicode characters.

Usage & Invocation - JSON UTF-8

The JSON parser has been enhanced to support text in UTF-8 encoding in addition to the current EBCDIC-1047.

The two encodings can not be comingled, you can not use both at the same time in the same JSON text.

The parser will attempt to autodetect the encoding and process the text appropriately. Alternatively the new service, HWTJSENC can be used to manually set the correct encoding. Another new service, HWTJGENC, can be used to retrieve the encoding used to parse the text.

No change is required to existing applications that deal with EBCDIC-1047 data.

© 2019 IBM Corporation

Usage & Invocation - JSON UTF-8 - HWTJGENC

The new HWTJGENC service can be used to learn what (if any) JSON text encoding has been asserted to/discovered by a designated parser instance.

When the parser instance is first initialized, HWTJINIT(), a call to HWTJGENC() would return HWTJ_ENCODING_UNKNOWN.

At this point the application may (optionally) assert one of the supported encodings:

- **HWTJ_ENCODING_UTF8**
- **HWTJ_ENCODING_EBCDIC**

Once the application invokes the parser, HWTJPARS(), the parser instance discovers the encoding of the input JSON text. If the type was unknown, or if the discovered type agrees with whatever type the user asserted, then parse processing proceeds. After successful parse, HWTJGENC() would indicate UTF8/EBCDIC as appropriate, and we would say that this encoding type is now “in effect” for the parser instance, remaining so for the life of the instance.

Non-REXX syntax:

```
CALL HWTJGENC(
        ReturnCode,
        ParserHandle,
        Encoding,
        DiagArea);
```

REXX syntax:

```
address hwjson “hwtjgenc”,
        “ReturnCode”,
        “ParserHandle”,
       “Encoding”,
       “DiagArea.”
```

Usage & Invocation - JSON UTF-8 - HWTJSENC

The new HWTJSENC service can be used to assert that the JSON text supplied to the current parser instance is encoded as either of:

- **HWTJ_ENCODING_UTF8**
- **HWTJ_ENCODING_EBCDIC**

This service can only be invoked after the parser handle instance is initialized via HWTJINIT(), but before any manipulation of the JSON text is done, HWTJPARS()/HWTJCREN().

Calling HWTJSENC() is typically optional: the JSON text supplied to HWTJPARS() will undergo "discovery" processing in any case.

- The user might elect to use HWTJSENC() just as a sanity check (since the parser instance will demand that the discovered encoding type agree with that which the user asserted).
- It should be used in "create from scratch" scenarios. If the user has not asserted an encoding prior to calling HWTJCREN, the parser may attempt discovery on the supplied value text, but that value may not be "discoverable". In such cases the parser instance will assume EBCDIC as the default.

Non-REXX syntax:

```
CALL HWTJSENC(
            ReturnCode,
            ParserHandle,
            Encoding,
            DiagArea);
```

REXX syntax:

```
address hwjson "hwtjsenc",
             "ReturnCode",
             "ParserHandle",
            "Encoding",
            "DiagArea."
```

Usage & Invocation - JSON UTF-8

New sample shipped in SYS1.SAMPLIB that takes advantage of the UTF-8 support:

- HWTJXRX3 (Rexx)

© 2019 IBM Corporation

Installation - JSON UTF-8

- Rolled back to z/OS 2.2 and z/OS 2.3 via OA56139
- IPL is required after applying the APAR

© 2019 IBM Corporation

Overview – JSON Shallow Search

- Who (Audience)
  - Toolkit z/OS JSON parser users
- What (Solution)
  - The JSON parser added support for a shallow search scope.
- Wow (Benefit / Value, Need Addressed)
  - Addition of a shallow scoping option greatly improve the performance cost of searching through JSON documents by providing the ability for an application to control the depth of its search.

  - Shallow search scoping also eliminates the possibility of mistakenly finding a like-named member that is outside the intended search scope of an object that intentionally omitted that named member.

Usage & Invocation - JSON Shallow Search

The application can restrict the depth of the search using the new **SearchType** value **HWTJ_SEARCHTYPE_SHALLOW** added to the existing HWTJSRCH service.

Similar to HWTJ_SEARCHTYPE_OBJECT, but with limited depth of search, a shallow search will not consider content within any nested object(s) of that object which defines the scope of the search. Arrays may not be shallow searched, and starting handles should not reference content within nested objects.

Usage & Invocation - JSON Shallow Search

Sample JSON:

```
{
  "a": "A1",
  "b": {
         "c": "C1",
         "d": {
                "c": "C2",
                "e": "E1",
                "f": "F1"
              },
          "e": "E2"
       },
  "c": "C3"
}
```

- The **descendants** of the root object are: a (with value A1), b (object), c (with value C1), d (object), c (with value C2), e (with value E1), f (with value F1), e (with value E2), and c (with value C3).
- The **direct children** of the root object are: a (with value A1), b (object), and c (with value C3).
- A search of type HWTJ_SEARCHTYPE_OBJECT for "a", under the root object, yields the handle of object entry a (with value A1), as would a search of type HWTJ_SEARCHTYPE_SHALLOW.
- A search of type HWTJ_SEARCHTYPE_OBJECT for "c", under the root object, yields the handle of object entry c (with value C1), whereas a search of type HWTJ_SEARCHTYPE_SHALLOW for "c", under the root object, yields the handle of c (with value C3).
- A search of type HWTJ_SEARCHTYPE_OBJECT for "d", "e", or "f", under the root object, yields the handle of d (object), e (with value E1), or f (with value F1), respectively, whereas a search of type HWTJ_SEARCHTYPE_SHALLOW for d, e, or f, under the root object, yields a HWTJ_JSRCH_SRCHSTR_NOT_FOUND return code form HWTJSRCH in each case.

Usage & Invocation - JSON Shallow Search

Sample JSON:

```
{
  "a": "A1",
  "b": {
         "c": "C1",
         "d": {
                "c": "C2",
                "e": "E1",
                "f": "F1"
              },
          "e": "E2"
       },
  "c": "C3"
}
```

Search for ”c” under the root object

```
rootObject = 0      /* search the root object */
startingHandle = 0  /* start with the first member */
entryName = ”c”
```

/* A search of type HWTJ_SEARCHTYPE_OBJECT for "c", under the root object, yields the handle of object entry c (with value C1) */

```
HWTJSRCH(returnCode, parserHandle, HWTJ_SEARCHTYPE_OBJECT,
         entryName, length(entryName), rootObject,
         startingHandle, targetEntryHandle, diagArea)
```

/* a search of type HWTJ_SEARCHTYPE_SHALLOW for "c", under the root object, yields the handle of c (with value C3) */

```
HWTJSRCH(returnCode, parserHandle, HWTJ_SEARCHTYPE_SHALLOW,
         entryName, length(entryName), rootObject,
         startingHandle, targetEntryHandle, diagArea)
```

© 2019 IBM Corporation

Installation - JSON Shallow Search

- Rolled back to z/OS 2.2 and z/OS 2.3 via OA56227
- IPL is required after applying the APAR

© 2019 IBM Corporation

Overview – JSON pretty print

- Who (Audience)
    - JSON Parser users
- What (Solution)
    - JSON text formatter
- Wow (Benefit / Value, Need Addressed)
    - The serialized output from the JSON parser is not formatted and is hard to read. The pretty print utility takes that output and formats it into a human readable style.

Usage & Invocation – JSON pretty print

The pretty print utility is a rexx program shipped in SYS1.SAMPLIB dataset and under the samples directory:

- SYS1.SAMPLIB(HWTJSPRT)
- /samples/jsonprint

The utility is limited to the encoding of the environment it’s running in, under z/OS it will only be able to format IBM-1047 encoded JSON content.

parserOutput.json

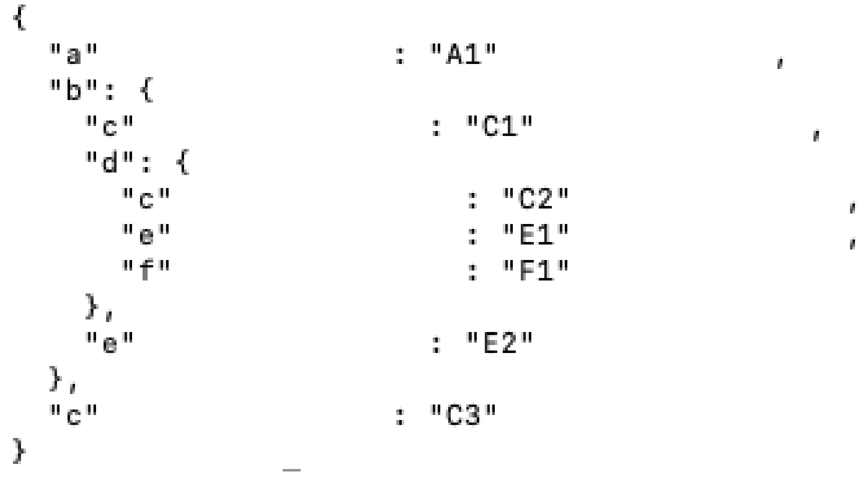
```
{"a": "A1","b":{"c": "C1","d": {"c": "C2","e": "E1","f": "F1"},"e": "E2"},"c": "C3"}
```

```
/* REXX */
call syscalls 'ON'
inputFile = ‘parserOutput.json'
outputFile = ‘pretty.json’

call bpxwunix '/samples/jsonprint’ inputFile ‘>’ outputFile

return
```

pretty.json



```
{
  "a"                 : "A1"                ,
  "b": {
    "c"                 : "C1"                ,
    "d": {
      "c"                 : "C2"                ,
      "e"                 : "E1"                ,
      "f"                 : "F1"
    },
    "e"                 : "E2"
  },
  "c"                 : "C3"
}
```

Installation – JSON pretty print

- Rolled back to z/OS 2.2 and z/OS 2.3 via OA55438

© 2019 IBM Corporation

Session Summary

z/OS 2.4 release of z/OS client web enablement toolkit includes enhancements for both the HTTP/HTTPS protocol enabler and the JSON parser.

- Applications that take advantage of the HTTP/HTTPS protocol enabler can now communicate with authenticating proxies, using either basic proxy authentication or taking advantage of AT-TLS support.
- Applications that use the JSON parser now have the ability delete entries, limit the depth of a search, use UTF-8 data directly, and format serialized IBM-1047 output.

Appendix

Publications

- **z/OS MVS Programming: Callable Services for High-Level Languages**
  - Complete toolkit documentation
- **z/OS MVS System Messages, Volume 6 (GOS – IEA)**
  - Toolkit message documentation
- **z/OS MVS System Codes**
  - Toolkit abend ’04D’x documentation

References

- TLS Handshake protocol
  https://tools.ietf.org/html/rfc5246#section-7.3
- HTTP CONNECT method (used to open a tunnel for HTTPS)
  https://tools.ietf.org/html/rfc7231#section-4.3.6
- The ‘Basic’ HTTP Authentication Scheme
  https://tools.ietf.org/html/rfc7617
- Proxy-Authorization header definition
  https://tools.ietf.org/html/rfc7235#section-4.4
- AT-TLS (top-level page)
  https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.3.0/com.ibm.zos.v2r3.halx001/transtls.htm

© 2019 IBM Corporation