Seriously? JSON parsing and REST API Calls From Your Legacy z/OS Applications? Absolutely!

Steve Warren
IBM
email: swarren@us.ibm.com
: @StevieWarr2

Agenda

- REST, JSON and HTTP
- Quick introduction to the toolkit
- z/OS HTTP/HTTPS protocol enabler details
  - Overview
  - Structure of an HTTP toolkit application
  - HTTP Connection Details
  - HTTP Request Details
  - Problem determination
- z/OS JSON Parser details
- Where to go for more information

2

REST, JSON, and HTTP

Moving Beyond the Browser

Businesses Are Evolving

Web APIs are the new, fast-growing business channel

Not having an API today is like not having a Web Site in the 90s

"***$7bn worth of items*** on eBay through APIs"
Mark Carges (Ebay CTO)

The API which has easily ***10 times more traffic*** then the website, has been really very important to us."
Biz Stone (Co-founder, Twitter)

"The adoption of Amazon's Web services is currently driving more network activity then everything Amazon does through their traditional web sites."
Jeff Bar (Amazon evangelist) / Dion Hinchcliffe (Journalist)

Web 1994 was the "get me a domain and a page" era.
Web 2000 was the "make my page(s) interactive and put people on it" era.
Web 2010 till now is the "get rid of pages and glue APIs and people together" era.
Robert Scoble (Author of tech blog Scobleizer)

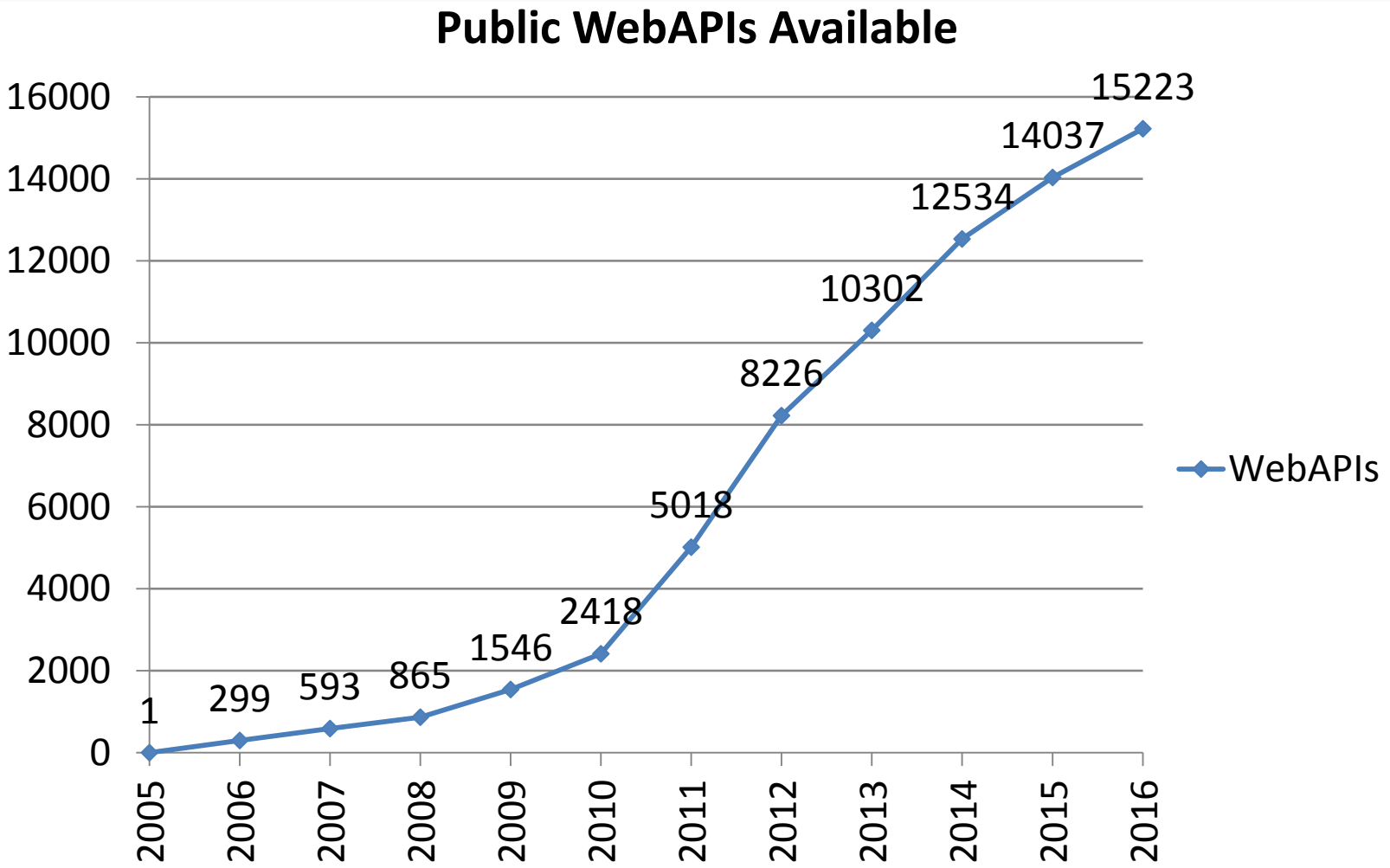
The API Economy – It's for real

"Today, application programming interfaces are the new must-have for business, representing the future of customer and community engagement with far broader implications than traditional Web-based business models." **Forbes 2012**

"The API is actually the driving force behind most of the **digital disruption** in the consumer space happening right now — cloud-based infrastructure, mobile apps, Facebook logins, online shopping and viewing — it's just that most laymen don't realize it. Indeed, APIs are surreptitiously increasing connectivity and enabling unprecedented services, disrupting the way we interact with the world." **Nordic APIs 2015**

"Today's leading enterprises are transforming digitally, jumping head first into the API economy. Transformations are being driven by connected devices and consumers' thirst for compelling brand experiences—all generating a vast and ever-growing amount of data." **IBM 2016**

Growth in Publically Published Web APIs

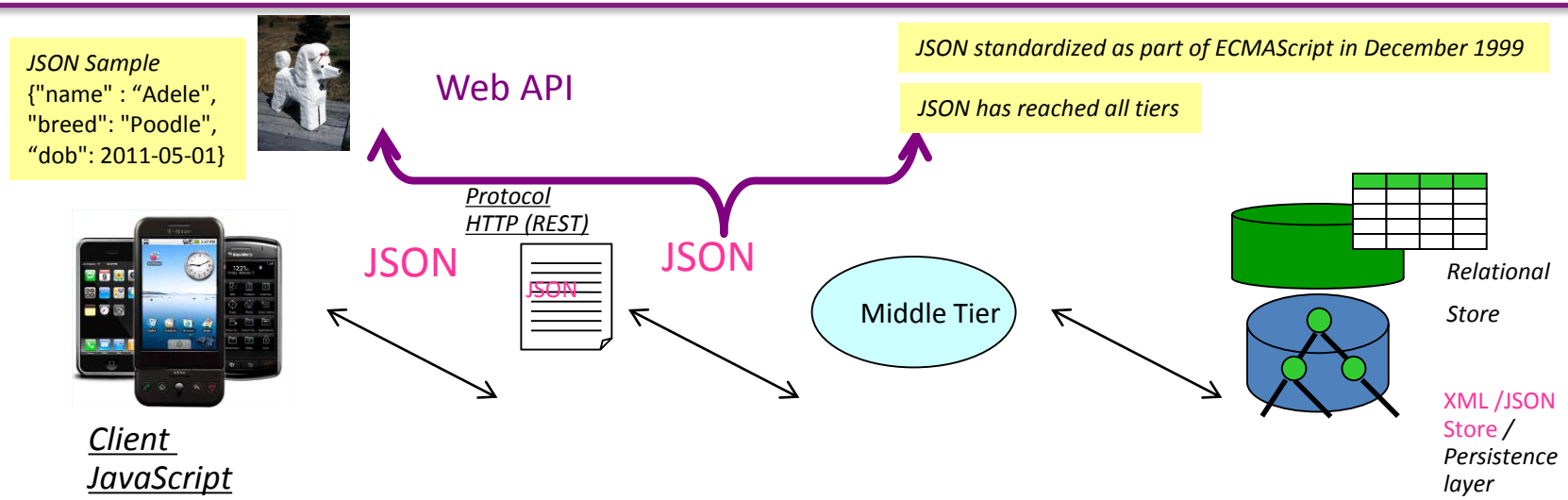


Source: Programmable Web (some intermediate numbers extrapolated)

What are REST APIs?

- Web applications are a client/server programming model using a request/response protocol.
- RESTful applications are web applications that follow a simple set of architectural constraints such as:
  - using standard HTTP methods
  - stateless servers
  - using URIs (URLs) strictly to identify the server resource being modified or interrogated
  - sending data back and forth in human-readable form
- REST APIs are defined by a server
  - What URI should be interrogated
  - Which HTTP service should be used against that URI
  - The format of the data to be sent back and forth

JSON : The Exchange Notation For Mobile Devices



- With the increased popularity of Web APIs (literally thousands of Web APIs) and the use of Mobile Devices
  - User Interfaces usually have a JavaScript component
    - JSON is the data structure for JavaScript
  - JSON is integrated with JavaScript and Java and other languages (through libraries)
  - The JSON trend is developer driven and is reaching all tiers (UI, Middle Tier, Data Tier)

Aspects of JSON:

No namespaces

No schemas

No mixed content support Mixed content example: hello Adelehow are you

JSON Penetration: Web API Trend Towards JSON

- “As more and more Web and mobile applications utilize APIs to drive their respective front ends, performance becomes an emerging concern. XML, long used as a method for exchanging data, is giving way to JSON, now considered the gold standard. ” - Programmable Web 2013
- “JSON's simplicity has made it a favored data exchange format” – Mashery 2014
- “In general, JSON wins the battle on brevity which is why many web applications are using JSON for RESTful data transfer.” – GCN 2014

JSON example

```
{
"firstName": "Steve",
"lastName": "Jones",
"age": 46,
"address": {
        "streetAddress": "123 Anywhere Ave",
        "city": "Poughkeepsie",
        "state": "NY",
        "postalCode": "12601",
        "country": "USA"
        },
"phone": [
        {
          "type": "mobile",
          "number": "914 555 5555"
        },
        {
          "type": "home",
          "number": "845 555 1234"
        }
    ]
}
```

Name – value pairs

- Name in quotes
- Value one of the types below

Objects – { }

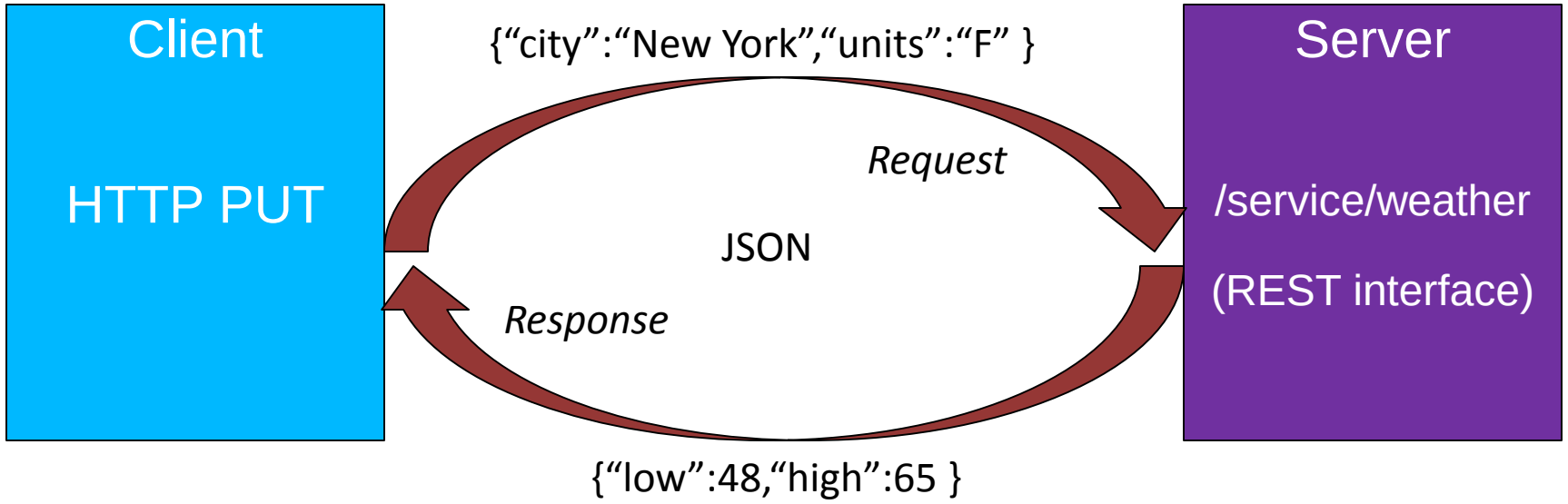
Arrays – [ ]

Object entries

- Other objects or arrays
- String – “ ”
- Numbers
- Boolean
- Null

10

30,000 foot view of REST / HTTP / JSON



z/OS serving REST APIs

- z/OS platform has for years been labeled “the server of servers” and houses much of the world’s most critical data.
- Enhancements to the z/OS Web serving space through the years have allowed this mammoth workhorse and repository of data to be more easily accessible to other systems.
- Options available to serve countless REST requests coming from just about anywhere into the z/OS mainframe world.
  - IBM HTTP Server powered by Apache
  - WebSphere* Liberty
  - WebSphere Classic
  - z/OS Connect

Intro to the toolkit

Introducing the z/OS Client Web Enablement Toolkit!

The z/OS client web enablement toolkit provides a set of lightweight application programming interfaces (APIs) to enable traditional, native z/OS programs to participate in modern web services applications.

- Pieces of the toolkit:
  - A z/OS HTTP/HTTPS protocol enabler to externalize HTTP and HTTPS client functions in an easy-to-use generic fashion for user's in almost any z/OS environment
  - A z/OS JSON parser which parses JSON coming from any source, builds new JSON text, or adds to existing JSON text.
- The toolkit allows its two parts to be used independently or combined together.
  - Payload processing is separate from communication processing.
- The interfaces are intuitive for people familiar with other HTTP enabling APIs or other parsers
- Easy for newbies

General programming toolkit environment

- Runs in just about any address space
  - Code runs in user's address space
- Supports both authorized and un-authorized callers
- Easy API suite provided
- Multi-language support
  - Include files supplied for C, COBOL, PL/I, Assembler
  - Multi-language samples provided

15

Recent toolkit announcement

- Customers asked numerous times…
  - We listened!
- Introducing…a new…
  - **REXX API interface to the toolkit for both:**
    - the z/OS JSON parser
    - the HTTP/HTTPS protocol enabler
  - An easy to use interface
  - APAR OA50659 will be delivered likely in the next few weeks

16

z/OS HTTP/HTTPS protocol enabler details

Usage & Invocation – z/OS HTTP Services

- Provides similar functionality to existing open-source libcurl HTTP/HTTPS interface
  - Interface is very similar
  - Underlying code is z/OS-specific and not ported in any way

18

HTTP features supported by toolkit

- HTTPS connections
- HTTP cookies management
- Proxies
- URI redirection
- Basic client authentication
- Chunked encoding
- Interoperability with AT-TLS
  - Coming soon via APAR OA50957

19

HTTP Services execution environment

- Specific requirements for the HTTP portion of the toolkit:
  - Supports task mode, non-cross-memory callers
  - Execution key 1 thru 15 allowed
  - OMVS segment required for address space using HTTP enabler
- Recovery recommended by caller

20

Structure of an HTTP toolkit application

Connections / Requests

- The HTTP/HTTPS enabler portion of the toolkit encompasses two major aspects of a web services application:
  - The ***connection*** to a server
  - The ***request*** made to that server along with the response it returns

HTTP Connections

- A connection is simply a socket (pipeline) between the application and the server.
- Must be established first before a request can flow to the server.
- Many options available for connection including:
  - SSL/TLS
  - Local IP address specification
  - IP Stack
  - Timeout values

Steps to create an HTTP Connection

- Initialize a connection (HWTHINIT)
  - Obtain workarea storage for the connection
- Set one or more connection options (HWTHSET)
  - One option at a time
- Make the actual connection (HWTHCONN)
  - Creates the socket to the specified server

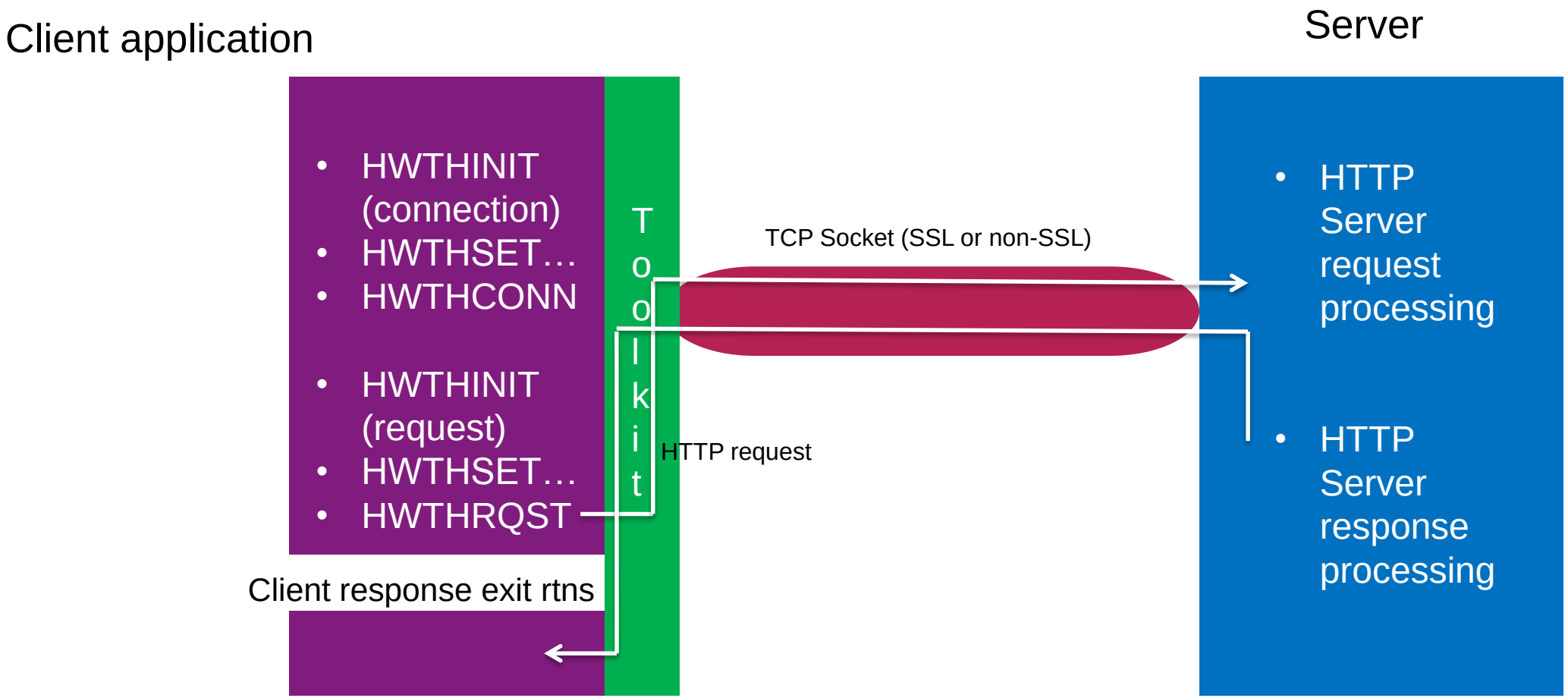
HTTP Requests

- An HTTP request sent over an existing connection
  - Targets a particular resource at the domain established by the connection
  - An HTTP GET, PUT, POST or DELETE is specified as the request method
- Requests not tightly-coupled to a connection. The same request can be sent over different connections
- Response callback routines (exits) can be set prior to the request to all processing of the response headers and response body.

Steps to create an HTTP Request

- Initialize a request (HWTHINIT)
  - Obtain workarea storage for the request
- Set one or more request options (HWTHSET)
  - One option at a time
- Send the request over a specified connection (HWTHRQST)
  - Flows the HTTP REST API call over the connection (socket) and then receives the response

Summary of connections and requests



HTTP Connection Details

Where do I want to connect?

- REST API specification details the location where the request must target
- Here are some examples:
  - Google® Maps Directions API specifies:
    - https://maps.googleapis.com/maps/api/directions/*outputFormat?paramaters*
  - Yelp® Search API specifies:
    - http://api.yelp.com/v2/search*?searchParms*
  - FAA Airport Service API specifies:
    - http://services.faa.gov/airport/status/*airportCode*

29

Format of a URI (URL) in the world of HTTP

- A Uniform Resource Identifier (URI) is the way a REST API is represented. A simplified syntax of a URI is:
  - **`scheme://host[:port]`** **`[/]path[?query][#fragment]`**

|---------- where and how to connect? Connection portion of URI ----------| |---------- what is the particular request? Request portion of URI ----------|

- scheme – HTTP (unencrypted) or HTTPS (encrypted) (optional)
- host – hostname or IPv4 or IPv6 address (e.g. www.ibm.com)
- port – an optional destination port number (80 is default for HTTP scheme, 443 for HTTPS scheme)
- path – an optional hierarchical form of segments (like a file directory) which represents the resource to perform the HTTP request method against
- query – an optional free-form query string to allow for the passing of parameters
- fragment – an optional identifier providing direction to a secondary resource

30

Setting Location of Connection

- HWTH_OPT_URI (required) – valid values are either a v4 or v6 IP address, or hostname
  - Examples:
    - http://192.168.0.1
    - http://[2001:1890:1112:1::20]
    - http://www.example.com
  - HTTP Scheme is optional
- HWTH_OPT_PORT (optional) – value specifying which remote port number to connect to, instead of the one specified in the URL or the default HTTP or HTTPS port.
  - Default for HTTP is 80
  - Default for HTTPS is 443

31

Setting Source Location of Connection

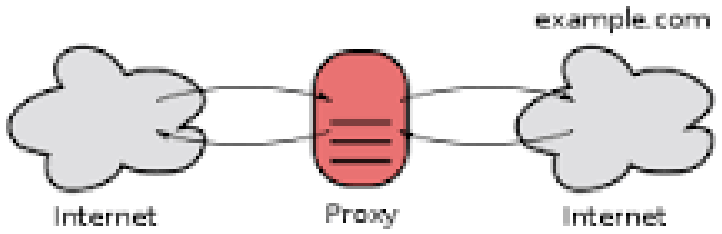
- HWTH_OPT_IPSTACK – Optional value 1 to 8 character z/OS TCP/IP stack to be used by the connection
  - Procname of TCP/IP (useful when installation has more than one TCP/IP stack running)
- HWTH_OPT_LOCALIPADDR – Optional outgoing local IP address
- HWTH_OPT_LOCALPORT – Optional outgoing local port

- Definitions also useful especially when security definitions are in place for specifically defined addresses and ports
  - Application limited by existing security profiles and definitions that are already in effect on the system where the application resides.
    - z/OS Communications Server NetAccess, in conjunction with security profiles defined using the SERVAUTH class, can be used to control network access authority, TCP/IP stack access authority, port access authority, and more.

32

HTTP Services – Proxy Options

- HWTH_OPT_PROXY – set the HTTP proxy to user. Specified the exact same as HWTH_OPT_URI above.
- HWTH_OPT_PROXYPORT – specify the proxy port to connect to. Specified the exact same as HWTH_OPT_PORT above



33

Setting other connection options

- HWTH_OPT_SNDTIMEOUTVAL – Sending timeout value
- HWTH_OPT_RCVTIMEOUTVAL – Receiving timeout value
- HWTH_OPT_HTTP_VERSION – which version of HTTP do you want to use over this connection
  - HTTP_VERSION_NONE
  - HTTP_VERSION_1_0
  - HTTP_VERSION_1_1

34

HTTP Services – SSL Options

- SSL support options include:
  - HWTH_OPT_USE_SSL – tells toolkit to attempt SSL negotiation explicitly
  - HWTH_OPT_SSLVERSION – sets the SSL versions to be supported by this HTTP request. More than one version may be selected. (e.g. TLS1.2, TLS1.1, TLS1.0, SSLv3)
  - HWTH_OPT_SSLKEYTYPE – Specifies the manner the key will be supplied to this HTTPS request. The following constants are provided:

    SSLKEYTYPE_KEYDBFILE – key store is specified key database file

    SSLKEYTYPE_KEYRINGNAME – key store is a security product managed keyring.
  - HWTH_OPT_SSLKEY – Specifies the value of the key. The value specified depends on the value set by SSLKEYTYPE.

    For SSLKEYTYPE_KEYDBFILE - represents path and name of the key database file name

    For SSLKEYTYPE_KEYRINGNAME – represents the SAF key ring name or PKCS#11 token
  - HWTH_OPT_SSLKEYSTASHFILE – specifies the stash file of the key database file. Only valid if SSLKEYTYPE_KEYDBFILE is specified. Ignored in all other cases.
  - HWTH_OPT_SSLCLIENTAUTHLABEL – optional label that represents a client certificate if SSL client authentication is requested by the server.

HTTP Services – AT-TLS / Toolkit Interoperability Coming

- Application Transparent – TLS is basically stack-based TLS
  - TLS process performed in TCP layer (via System SSL) without requiring any application change (transparent)
  - AT-TLS policy specifies which TCP traffic is to be TLS protected based on a variety of criteria
    - Local address, port
    - Remote address, port
    - z/OS userid, jobname
    - Time, day, week, month
- The toolkit plans to provide AT-TLS interoperability support via APAR OA50957
  - Toolkit will add new HWTH_OPT_USE_SSL modes to determine its behavior in relationship to SSL
    - One mode will allow the connection to be upgraded to SSL/TLS and to learn of the upgrade and act accordingly (now knowing that AT-TLS has upgraded the connection)
    - Another mode will tolerate an upgrade provided the upgraded connection meets given criteria
    - Another mode will see if AT-TLS has upgraded the connection. If not, the application will specify the necessary upgrades

AT-TLS

Statements regarding IBM future direction and intent are subject to change or withdrawal, and represent goals and objectives only.

HTTP Services – Redirect Options

- Redirect options include:
  - HWTH_OPT_MAX_REDIRECTS – maximum number of redirects to follow for a given request.
  - HWTH_OPT_XDOMAIN_REDIRECTS – are cross-domain redirects allowed?
  - HWT_OPT_REDIRECT_PROTOCOLS – do you allow the HTTP/HTTPS protocol to be upgraded and/or downgraded on a redirect?

37

HTTP Services – Cookie Options

- Cookie support options include:
  - HWTH_OPT_COOKIETYPE – sets cookie handling type

    COOKIETYPE_NONE – cookie engine not activated

    COOKIETYPE_SESSION – cookie engine enabled – cookies automatically sent, but end when connection ends

    COOKIETYPE_PERSIST - cookie engine enabled – cookies automatically sent, cookies saved to output buffer when connection endsin
  - HWTH_OPT_COOKIE_INPUT_BUFFER – specifies input cookie data store
  - HWTH_OPT_COOKIE_OUTPUT_BUFFER – specifies output cookie location for a cookietype of COOKIETYPE_PERSIST

Example of initializing a connection

```
HandleType = HWTH_HANDLETYPE_CONNECTION
address hwthttp "hwthinit ",
                "ReturnCode ",
                "HandleType ",
                "ConnectHandle ",
                "DiagArea."
```

- A connection instance has been created.
- More than one connection can be initialized per address space (if the user has set his dubbing defaults to dub process)
  - But only one connection can be active at a time (Setup MVS Signals only allows one signal setup to be allowed per process)

39

Example of setting some connection options

```
/*************************************************************/
/* Set URI for connection to the Federal Aviation Administration (FAA) */
/*************************************************************/
ConnectionUri = 'http://services.faa.gov'
ReturnCode = -1
DiagArea. = ''
address hwthttp "hwthset ",
                "ReturnCode ",
                "ConnectHandle ",
                "HWTH_OPT_URI ",
                "ConnectionUri ",
                "DiagArea."
/*************************************************************************************/
/* Set HWTH_OPT_COOKIETYPE                                                           */
/*   Enable the cookie engine for this connection.                                   */
/*   Any "eligible" stored cookies will be resent to the host on subsequent interactions automatically. */
/*************************************************************************************/
ReturnCode = -1
DiagArea. = ''
address hwthttp "hwthset ",
                "ReturnCode ",
                "ConnectHandle ",
                "HWTH_OPT_COOKIETYPE ",
                "HWTH_COOKIETYPE_SESSION ",
                "DiagArea."
```

- All the connection options can be set prior to the connect service
- Once the connect services is issued, most set option calls will have no effect until the connection is disconnected and reconnected again.

40

Example of finally connecting to the HTTP server

```
/*********************************/
/* Call the HWTHCONN toolkit api  */
/*********************************/
ReturnCode = -1
DiagArea. = ''
address hwthttp "hwthconn ",
                "ReturnCode ",
                "ConnectHandle ",
                "DiagArea."
```

- The connection stays persistent from a toolkit perspective
  - Timed-out connections (sockets) will automatically be reconnected at the time of a request if necessary
- A disconnect, reset or terminate of the connection will disconnect the established connection

41

HTTP Request Details

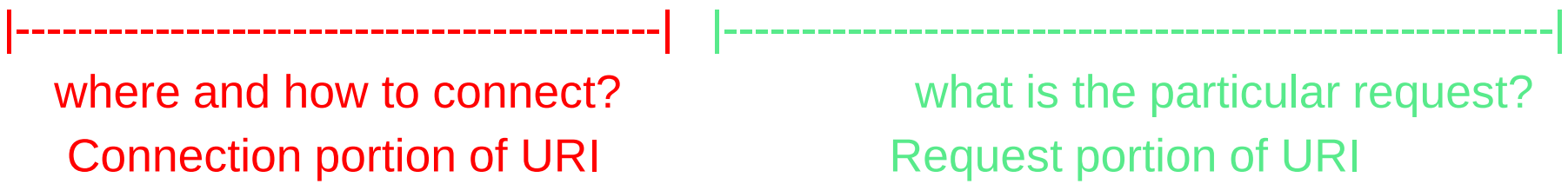
What resource and parameters do I want to invoke?

- REST API specification details the location where the request must target
- Here are some examples:
  - Google® Maps Directions API specifies:
    - https://maps.googleapis.com/maps/api/directions/*outputFormat?paramaters*
  - Yelp® Search API specifies:
    - http://api.yelp.com/v2/search*?searchParms*
  - FAA Airport Service API specifies:
    - http://services.faa.gov/airport/status/*airportCode*

43

Format of a URI (URL) in the world of HTTP

- A Uniform Resource Identifier (URI) is the way a REST API is represented. A simplified syntax of a URI is:
  - **`scheme://host[:port]`** `[/]path[?query][#fragment]`



- scheme – HTTP (unencrypted) or HTTPS (encrypted) (optional)
- host – hostname or IPv4 or IPv6 address (e.g. www.ibm.com)
- port – an optional destination port number (80 is default for HTTP scheme, 443 for HTTPS scheme)
- path – an optional hierarchical form of segments (like a file directory) which represents the resource to perform the HTTP request method against
- query – an optional free-form query string to allow for the passing of parameters
- fragment – an optional identifier providing direction to a secondary resource

Setting request path, input parameters, method

- HWTH_OPT_URI (required) – The name or resource (URN path portion) of the URI. The query and fragment portions of a URI may also be present.
  - Examples:
    - /systems/z/
    - /over/here?name=abc#frag1
- HWTH_OPT_REQUEST (required) – which HTTP CRUD request method does the request want to use
  - HWTH_HTTP_REQUEST_GET
  - HWTH_HTTP_REQUEST_PUT
  - HWTH_HTTP_REQUEST_POST
  - HWTH_HTTP_REQUEST_DELETE

45

HTTP Headers

- HTTP header fields provide required information about the request or response, or about the object sent in the message body.
- The toolkit does not require you to send any request headers or process the response headers, if you don’t want to.
  - However, some servers demand that certain headers are sent. If the default headers that the toolkit sends are not sufficient for the server, then that header(s) will need to be sent
- Request headers are created by using the general purpose HWTHSLST service, which creates a linked list of objects together. After all the headers have been added, the HWTH_OPT_HTTPHEADERS option can be set, specifying the SLST created.

HTTP Request Headers

- Request headers are created by using the general purpose HWTHSLST service, which creates a linked list of headers, one linked list element added to the list per HWTHSLST call.
- After all the headers have been added, the HWTH_OPT_HTTPHEADERS option can be set, specifying the SLST created.

Response Headers

- In non-REXX languages, each response header sent from the server can be interrogated and/or processed by the response header callback (exit) routine.
  - Optionally set the HWTH_OPT_RESPONSEHDR_EXIT option to specify a 4-byte address of the exit to receive control
  - Optionally set the HWTH_OPT_RESPONSEHDR_USERDATA to pass a 4-byte address to the exit as userdata.

Response Headers

- In REXX, all the HTTP response headers are stored in a REXX stem variable as specified by setting the optional HWTH_OPT_RESPONSEHDR_USERDATA option.
- Upon return from the HWTHRQST (send request service), the following information is returned regarding the response headers received:
  - the number of response headers received is stored in the .0 stem
  - the name of each response header is stored in the stemname.x stem variable, where x is the from 1 to the number of response headers received.
  - the value for each response header is stored in the stemname.x.1 stem variable, where is x is from 1 to the number of response headers received.

Request Body

- The main data sent on a PUT or POST method to an HTTP REST server is usually sent in the request body
- In non-REXX, the HWTH_OPT_REQUESTBODY option specifies a 4-byte address of a buffer to be sent. The toolkit keeps a binding to the specified address until the HWTHRQST service has completed.
- In REXX, a simple name of a variable is set for the HWTH_OPT_REQUESTBODY option that sets the variable where the response body is stored

Response Body

- The main data returned from an HTTP REST server is usually sent as the response body
- In non-REXX, the response body callback (exit) routine will be given control once the entire response body has been received. The address of the callback routine is set by specifying a 4–byte address for the HWTH_OPT_RESPONSEBODY_EXIT option.
- An optional HWTH_OPT_RESPONSEBODY_USERDATA specifies a 4-byte value for an input parameter to the exit.
- The body exit can process this data and return back to the toolkit when completed.
- In REXX, the optional HWTH_OPT_RESPONSEBODY_USERDATA option is set to specify the name of a simple name of a variable where the response body is stored

HTTP Services – Chunked encoding

- **Responses with chunked encoding present**
  - Toolkit supports the chunked encoding data transfer method (Transfer-encoding: chunked).
  - Automatically de-chunks data sent from the server using the chunked encoding method.
    - The response body exit does not need to handle the various chunks; rather, the data is delivered to the exit already decoded.
    - If the chunked data contains trailer headers, the header exit will be invoked (once for each trailer header) prior to this routine receiving control.
  - Note: The toolkit ignores chunk extensions

52

Other HTTP request options

- HTTP authorization options:
  - HWTH_OPT_HTTPAUTH – Do I want HTTP basic client authentication?
  - HWTH_OPT_USERNAME and HWTH_OPT_PASSWORD must be set if basic client authentication is selected.
- HTTP request body and response body translate functions
  - HWTH_OPT_TRANSLATE_REQBODY – translate request body from EBCDIC to ASCII automatically.
  - HWTH_OPT_TRANSLATE_RESPBODY – translate response body from ASCII to EBCDIC automatically.

53

Example of initializing a request

```
HandleType = HWTH_HANDLETYPE_REQUEST
address hwthttp "hwthinit ",
                "ReturnCode ",
                "HandleType ",
                "ReqHandle ",
                "DiagArea."
```

- A request instance has been created.
- As many request instances as you would like can be created
- A request is married with a connection at the HWTHRQST API call

54

Example of setting some request options

```
/*************************************************************/
/* Set HTTP Request method.                                  */
/* A GET request method is used to get data from the server.  */
/*************************************************************/
address hwthttp "hwthset ",
                "ReturnCode ",
                "ReqHandle ",
                "HWTH_OPT_REQUESTMETHOD ",
                "HWTH_HTTP_REQUEST_GET ",
                "DiagArea."
*************************************************************/
* Set the request URI                                       */
*  Set the URN URI that identifies a resource by name that is   */
*    the target of our request.                             */
*************************************************************/
requestPath = '/airport/status/'||airportCode
address hwthttp "hwthset ",
                "ReturnCode ",
                "ReqHandle ",
                "HWTH_OPT_URI ",
                "requestPath ",
                "DiagArea."
```

- All the request options can be set before issuing the HWTHRQST service
- Once the request is completed, the request options can be altered if desired before next send request service.

55

Example of setting a request header

```
acceptJsonHeader = 'Accept:application/json'
/*****************************************************************
/* Create a brand new SList and specify the first header to be an     *
/* "Accept" header that requests that the server return any response  *
/* body text in JSON format.                                          *
/*****************************************************************
address hwthttp "hwthslst ",
                "ReturnCode ",
                "ReqHandle ",
                "HWTH_SLST_NEW ",
                "SList ",
                "acceptJsonHeader ",
                "DiagArea.“
/*************************************************/
/* Set the request headers with the just-produced list */
/*************************************************/
address hwthttp "hwthset ",
                "ReturnCode ",
                "ReqHandle ",
                "HWTH_OPT_HTTPHEADERS ",
                "SList ",
                "DiagArea."
```

- After the first HWTH_SLST_NEW function, specify HWTH_SLIST_APPEND to add more headers to the SLST if desired.

56

Example of setting more request options

```
/*************************************************************/
/* Tell the toolkit to translate the body from ASCII to EBCDIC  */
/*************************************************************/
address hwthttp "hwthset ",
                "ReturnCode ",
                "ReqHandle ",
                "HWTH_OPT_TRANSLATE_RESPBODY ",
                "HWTH_XLATE_RESPBODY_A2E ",
                "DiagArea."
/*************************************************/
/* Set the variable for receiving response body  */
/*************************************************/
say 'Set HWTH_OPT_RESPONSEBODY_USERDATA for request'
address hwthttp "hwthset ",
                "ReturnCode ",
                "ReqHandle ",
                "HWTH_OPT_RESPONSEBODY_USERDATA ",
                "ResponseBody ",
                "DiagArea."
```

- When the HWTHRQST service completes, the variable ResponseBody will have the response in EBCDIC.

57

Example of finally issuing the REST API call

```
/*************************************/
/* Call the HWTHRQST toolkit api.  */
/*************************************/
address hwthttp "hwthrqst ",
                "ReturnCode ",
                "ConnHandle ",
                "ReqHandle ",
                "HttpStatusCode ",
                "HttpReasonCode ",
                "DiagArea."
```

- When this API call completes, the request and response have also completed. The final status of the call is returned in the HttpStatusCode (eg. 200) and HttpReasonCode (eg. “OK”)
- Consult the other REXX variables set in the request after this call completes for response headers and response body (if-any).
- A reset or terminate of the request will wipe out any set values

58

Problem determination – HTTP Enabler

- **Return Code from service**
  - Specific return code can give explanation
- **DiagArea**
  - Many times provides detailed explanation.
- **Status values returned in callback routines**
  - Provides HTTP status values from server
- **HWTH_VERBOSE set option**
  - Toolkit directs many trace-like messages to the standard output of the application. Useful during debugging.
  - Can be directed to standard output or to a preallocated MVS data set or zFS file thru use of the HWTH_OPT_VERBOSE_OUTPUT option.
- **SOCKAPI CTRACE option**
- **System SSL tracing**

59

Example of turning on verbose tracing

```
address hwthttp "hwthset ",
                "ReturnCode ",
                "ConnHandle ",
                "HWTH_OPT_VERBOSE ",
                "HWTH_VERBOSE_ON ",
                "DiagArea."

traceDD = 'MYTRACE'
/* Allocate the data set here */
address hwthttp "hwthset ",
                "ReturnCode ",
                "ConnectionHandle ",
                "HWTH_OPT_VERBOSE_OUTPUT ",
                "traceDD ",
                "DiagArea."
```

- The DD name above must represent either:
  - a pre-allocated traditional z/OS data set which is a physical sequential (DSORG=PS) with a record format of unblocked variable (RECFM=V) or Undefined (RECFM=U) and expandable (non-zero primary and secondary extents). The DD must also specify a DISP=OLD disposition.
  - a zFS or HFS file.
- When a connect, sendRequest or disconnect are issued, detailed trace will be written to the data set specified by the MYTRACE DD.

60

z/OS Client Toolkit HTTP Language Support

Include files and sample programs provided in:

- C
- COBOL – sample delivered via APAR OA49002
- PL/I – sample delivered via APAR OA49002
- Assembler (Include file only)
- REXX – sample delivered via APAR OA50659

61

Installation – z/OS HTTP Enabler Services

- V2R2 – In the base
- V2R1 – Install APAR recommended below and re-IPL
- External message to know the toolkit is installed and ready to go:
  - HWT001I message will appear in the syslog stating the toolkit is enabled
- **Recommended: Install new APAR OA50586 toolkit fixpack #2 to pick up latest fixes**

62

z/OS JSON Parser details

z/OS JSON parser environment

- z/OS Client Toolkit execution environment:
  - Supports both authorized and un-authorized callers
    - Allow supervisor or problem state callers running in any PKM
  - Supports task and SRB mode invokers
  - Supports cross-memory mode invokers
  - Recovery needed by caller

Usage of the z/OS JSON parsing services

- How to use the services:
  - Initialize a parse instance
    - Returns a parser handle
  - Parse some JSON Text
  - Use traversal or search methods
    - Quick access to various constructs in the JSON text or to find a particular name
  - Re-use the parse instance or terminate it

65

z/OS JSON parsing services - Traverse

Traversal services include:

- Get JSON Type (HWTJGJST)
- Get Value (for string or numeric) (HWTGVAL)
- Get Numeric Value (HWTJGNUV)
- Get Boolean Value (HWTJBOV)
- Get Number of Entries (HWTJGNUE)
- Get Object Entry (HWTJGOEN)
- Get Array Entry (HWTJGAEN)

66

z/OS JSON parsing services - Search

- JSON search (HWTJSRCH):
  - Allows a particular “name” to be quickly consulted within the entire JSON text or within a particular object.
  - The value handle returned references the value associated with the found “name”
  - Two search types:
    - HWTJ_SEARCHTYPE_GLOBAL
    - HWTJ_SEARCHTYPE_OBJECT

67

z/OS JSON parsing services - Create

- JSON creation services:
  - Create JSON Entry (HWTJCREN)
  - Serialize JSON Text (HWTJSERI)
- Allows the creation of new JSON text or the addition of entries to existing JSON text.
- Provides option to merge multiple JSON text streams easily and to validate that the insertion point is syntactically valid
- Allows JSON text to be traversed even after new text added

z/OS Client Toolkit JSON Language Support

Include files and sample programs provided in:

- C/C++
- COBOL
- PL/I
- Assembler (Include file only)
- REXX (coming soon)

69

JSON Parsing Example

```
hwtjinit(&return_code,
    MAX_WORKAREA_SIZE,
    parser_instance,
    &diag_area);

if return_code != HWTJ_OK
  return -1;

hwtjpars(&return_code,
     parser_instance,
     (char *)&jtext, /* JSON text string address(input) */
     strlen(jtext),  /* JSON text string length (input) */
     &diag_area);
```


70

JSON Parsing Example (continued)

```
 hwtjsrch(&return_code,
     parser_instance,
     HWTJ_SEARCHTYPE_OBJECT, /* limit the search scope */
     (char *)&name,          /* search string address */
     strlen(name),           /* search string length */
     object_to_search,       /* handle of object to search */
     0,                      /* starting point of the search */
     &value_handle,          /* search result handle (output) */
     &diag_area);

/* Check that the search found a result. */
if (return_code == HWTJ_OK) {
  /* Get the object's type. */
  hwtjgjst(&return_code,
      parser_instance,
      value_handle, /* handle to the value whose type to check (input) */
      &entry_type, /* value type constant returned by hwtjgjst (output) */
      &diag_area);
```


71

Installation and Availability

- V2R2 – In the base
- V2R1 – Install latest toolkit APAR listed below and re-IPL
- External message to know the toolkit is installed and ready to go:
  - HWT001I message will appear in the syslog stating the toolkit is enabled

- **Recommended: Install new APAR OA50865 toolkit fixpack #2 to pick up latest fixes**

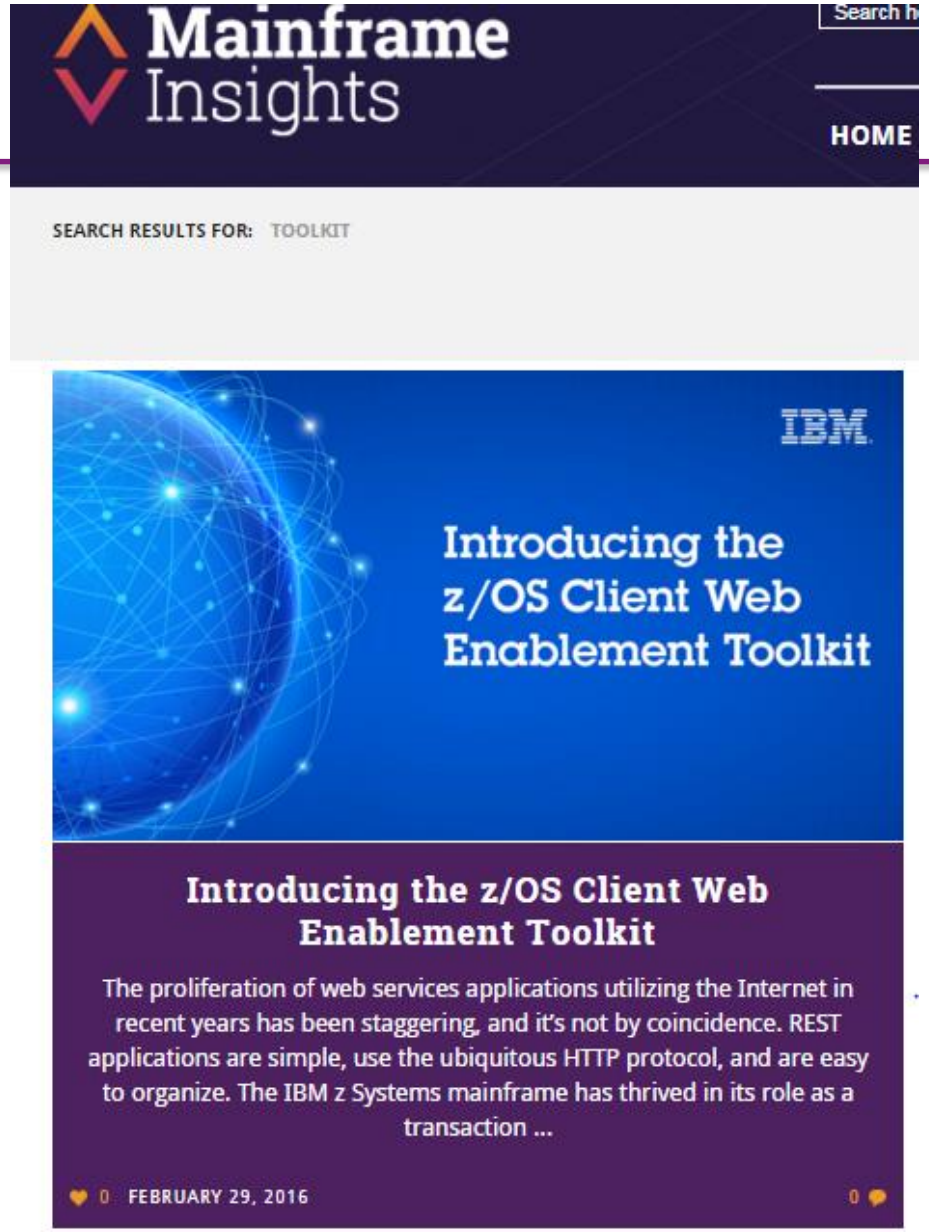
72

Where to go for more z/OS Client Web Enablement Toolkit information

Toolkit Reference Materials

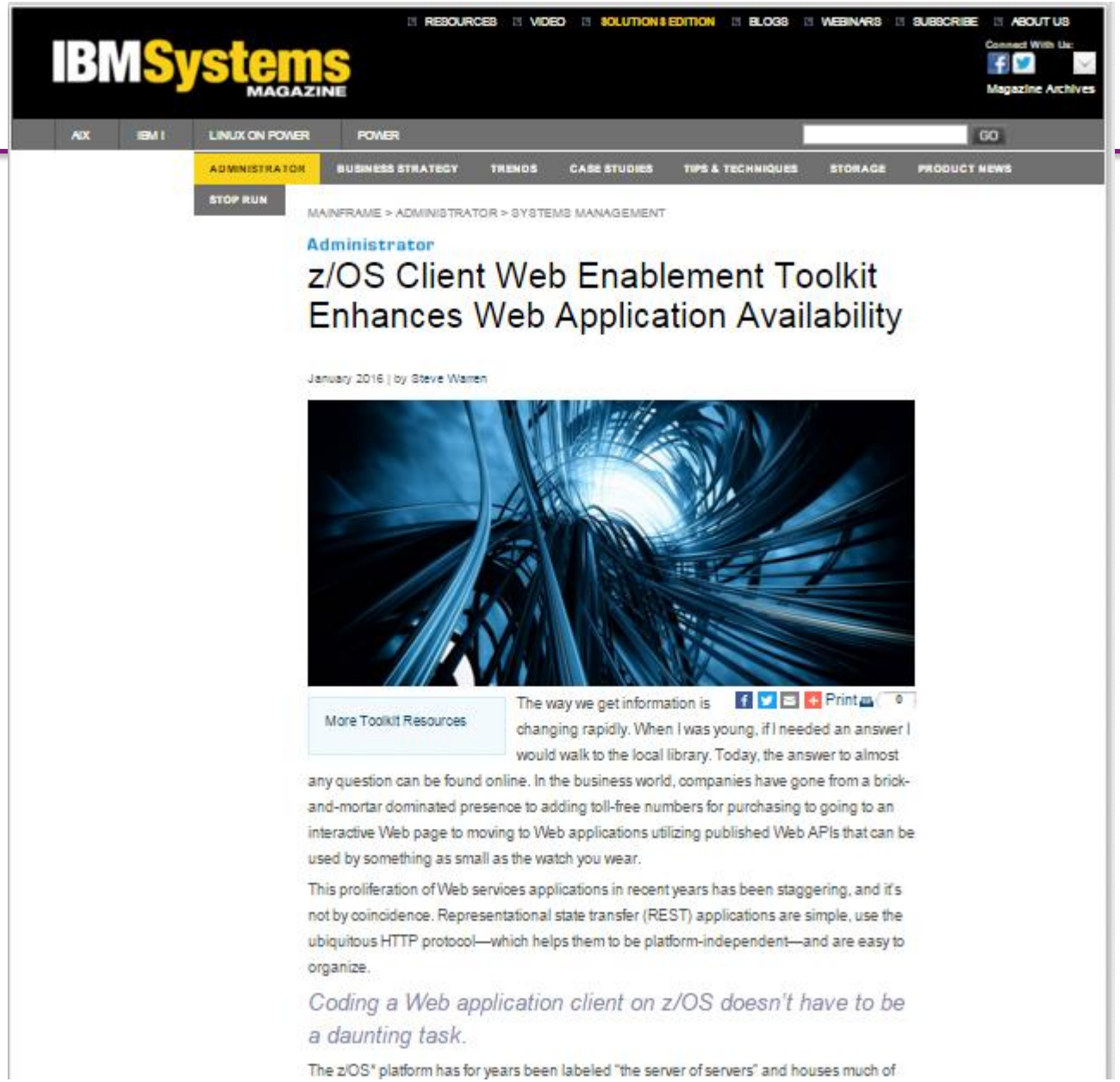
- **z/OS 2.2 MVS Programming: Callable Services for High-Level Languages**
  - Complete toolkit documentation
- **z/OS 2.2 MVS System Messages, Volume 6 (GOS – IEA)**
  - Toolkit message documentation
- **z/OS 2.2 MVS System Codes**
  - Toolkit abend ’04D’x documentation

74



- **mainframeinsights.com**
  - Search for toolkit

75



IBMSystems MAGAZINE

RESOURCES | VIDEO | SOLUTION S EDITION | BLOGS | WEBINARS | SUBSCRIBE | ABOUT US

Connect With Us:

Magazine Archives

AIX | IBM I | LINUX ON POWER | POWER | GO

ADMINISTRATION | BUSINESS STRATEGY | TRENDS | CASE STUDIES | TIPS & TECHNIQUES | STORAGE | PRODUCT NEWS

STOP RUN

MAINFRAME > ADMINISTRATOR > SYSTEMS MANAGEMENT

Administrator

z/OS Client Web Enablement Toolkit Enhances Web Application Availability

January 2016 | by Steve Warren

More Toolkit Resources

Print 0

The way we get information is changing rapidly. When I was young, if I needed an answer I would walk to the local library. Today, the answer to almost any question can be found online. In the business world, companies have gone from a brick-and-mortar dominated presence to adding toll-free numbers for purchasing to going to an interactive Web page to moving to Web applications utilizing published Web APIs that can be used by something as small as the watch you wear.

This proliferation of Web services applications in recent years has been staggering, and it's not by coincidence. Representational state transfer (REST) applications are simple, use the ubiquitous HTTP protocol—which helps them to be platform-independent—and are easy to organize.

Coding a Web application client on z/OS doesn't have to be a daunting task.

The z/OS® platform has for years been labeled "the server of servers" and houses much of

- **IBM Systems Magazine (January / February 2016)**
  - z/OS Client Web Enablement Toolkit Enhances Web Application Availability (pg 34 -36 in hardcopy edition)

REST easy on z/OS

Introducing the z/OS Client Web Enablement Toolkit

BY STEVE WARREN

The number of web service applications on the internet has increased significantly in recent years. RESTful applications that use HTTP or HTTPS as a means of communication and send JSON or XML data is as common as it gets in the mobile, client/server world.

Wouldn't it be cool if your existing z/OS applications running in a traditional environment could easily ramp up to play in the game as well as through a set of base z/OS services available to most programs on z/OS?

If you're excited about these options, welcome to the new z/OS Client Web Enablement Toolkit! Built into the base of the z/OS operating system, the toolkit provides a lightweight solution to enable these applications to more easily participate in this client/server space by providing the following built-in features:

- A z/OS JSON parser that can be used to parse JSON text that comes from any source and create new JSON text or add to existing JSON text.
- A z/OS HTTP/HTTPS protocol enabler which uses interfaces similar to other industry-standard APIs.

Just about all environments on z/OS can avail themselves of these new services. Traditional z/OS programs that run in native z/OS have little or no options that they can easily use to participate in web services applications. Programs running as a batch job, as a started procedure or in almost any address space on a z/OS system now have APIs that can be used in a similar manner to any standard z/OS APIs provided by the operating system.

Furthermore, programs can use these APIs in the programming language of their choice. You can use C/C++, COBOL, PL/I, and Assembler languages, and samples are provided for C/C++, COBOL, and PL/I.

Would you like to hear more about the parts of the toolkit and get a small taste of what you can do?

z/OS JSON parser

Suppose that you would like to be able to make sense of a large JSON text file that was sent to you from a web server that you are communicating with. The new z/OS JSON parser can do the heavy lifting for you.

The following questions can help you decide which style of parsing is best for you:

Welcome to the new z/OS Client Web Enablement Toolkit!

- Do you know the format of the data that is being returned?
- Are you looking for specific fields in a particular format?
- Do you need to learn about all the data that is returned?

Based on your answer, you can choose the "search" style, the "traversal" style, or a combination of both. The "search" style looks for specific key values in the text stream and then finds the values that are associated with those key values. The "traversal" style starts the traversal parser services to recursively move through the text stream until it learns what was sent.

In either case, a program that uses the JSON parsing services follows this format:

1. Start the JSON parser initialize service (HWTJINIT) to create a parsing instance.
   Tip: You can go wild here and create as many parser instances as your application requires. Each parser instance allows the z/OS JSON parser the ability to separately manage the parsing of a JSON text stream. The more instances you have, the more concurrent JSON text streams you can parse.
2. Call the JSON Parse service (HWTJPARS) to have the parser validate the syntax of the text stream and create an internal representation of the JSON text data. Once the data is parsed, it allows all subsequent services to run faster

- **z/OS Hot Topics magazine (August 2015)**
  - **REST easy on z/OS – Introducing the z/OS Client Web Enablement Toolkit** (pg. 26-27)

Questions?

78

Notices and Disclaimers

U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IN NO EVENT SHALL IBM BE LIABLE FOR ANY DAMAGE ARISING FROM THE USE OF THIS INFORMATION, INCLUDING BUT NOT LIMITED TO, LOSS OF DATA, BUSINESS INTERRUPTION, LOSS OF PROFIT OR LOSS OF OPPORTUNITY. IBM products and services are warranted according to the terms and conditions of the agreements under which they are provided.

IBM products are manufactured from new parts or new and used parts. In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply."

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer is in compliance with any law

Notices and Disclaimers Con't.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of 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. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. IBM EXPRESSLY DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The provision of the information contained h erein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, ibm.com, Aspera®, Bluemix, Blueworks Live, CICS, Clearcase, Cognos®, DOORS®, Emptoris®, Enterprise Document Management System™, FASP®, FileNet®, Global Business Services ®, Global Technology Services ®, IBM ExperienceOne™, IBM SmartCloud®, IBM Social Business®, Information on Demand, ILOG, Maximo®, MQIntegrator®, MQSeries®, Netcool®, OMEGAMON, OpenPower, PureAnalytics™, PureApplication®, pureCluster™, PureCoverage®, PureData®, PureExperience®, PureFlex®, pureQuery®, pureScale®, PureSystems®, QRadar®, Rational®, Rhapsody®, Smarter Commerce®, SoDA, SPSS, Sterling Commerce®, StoredIQ, Tealeaf®, Tivoli®, Trusteer®, Unica®, urban{code}®, Watson, WebSphere®, Worklight®, X-Force® and System z® Z/OS, are trademarks of International Business Machines Corporation, 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.

80

Thank you!