# z/OS V2.5 IBM Education Assistant

Solution Name: REST-enabled BCPii

Solution Element(s): z/OS BCP

August 2021

# Agenda

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

# Trademarks

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

# Objectives

- Present information about a new z/OS BCPii API that unlocks additional IBM Z hardware functions/attributes for BCPii applications

# Overview

- Who (Audience)
  - z/OS BCPii customers

- What (Solution)
  - A new z/OS BCPii API that allows application to access additional available IBM Z hardware attributes and functionality

- Wow (Benefit / Value, Need Addressed)
  - Instantly access additional hardware functions/attributes that are currently available and have immediate access to additional future hardware functions/attributes
  - A new API that provides a simpler and more intuitive REST programming model for application to leverage

# Usage & Invocation

A new z/OS BCPii API that accepts REST API's defined in Chapter 11. Core IBM Z resources chapter of *IBM Z Hardware Management Console Web Services API* Publication (Library → z15 → Web Services API)

- Same internal transport to the SE
  - Does NOT use the TCP/IP network for communications

- We refer to this new way of invoking the hardware **BCPii V2** throughout the rest of this presentation

Automatic access to new attributes

Single documentation reference for attributes

RestLIKE interface

– URI, Queryparms, Request body, Response Body

– JSON format

Easier command interface

# Usage & Invocation

New CPC attributes you'll be able to access:

- dpm-enabled
- is-cpacf-enabled
- is-secure-execution-enabled
- is-global-key-installed
- is-host-key-installed
- global-primary-key-hash
- global-secondary-key-hash
- host-primary-key-hash
- host-secondary-key-hash
- is-on-off-cod-installed
- has-temporary-capacity-change-allowed
- lan-interface1-type
- lan-interface1-address
- lan-interface2-type
- lan-interface2-address

- storage-total-installed
- storage-hardware-system-area
- storage-customer
- storage-customer-central
- storage-customer-expanded
- storage-customer-available
- storage-vfm-increment-size
- storage-vfm-total
- cpc-power-rating
- cpc-power-consumption

  *…additional power saving settings*
- zcpc-ambient-temperature
- zcpc-exhaust-temperature
- zcpc-humidity
- zcpc-dew-point
- zcpc-heat-load

  *…lots of other environmentals*

# Usage & Invocation

New LPAR attributes you'll be able to access:

- More detailed information about processors and their assignments to an LPAR
- Detailed information about storage allocated for the LPAR

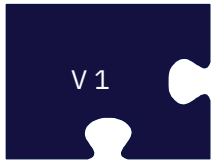
New commands/actions you'll be able to access:

- NVMe Load
- NVMe Dump
- Ability to get detailed resource allocations for LPARs on a single CPC in one directive

# Usage & Invocation – v1 vs v2

Comparison of existing API's (v1) to new RESTlike API(v2)

- In v2, a single API will be used to perform all the requests, from query to command




- HWICONN
- HWIDISC
- HWILIST
- HWIQUERY
- HWISET
- HWISET2
- HWICMD
- HWICMD2
- HWIEVENT
- z/OS USS Event delivery services


- HWIREST

Stand alone service that does not require prior invocation of any v1 services. In other words, this does not use the concept of the 'connect' token.

# Usage & Invocation – v1 vs v2 data format

V 1

IBM-1047 encoded plain text data or XML data (temporary capacity command)

V 2

IBM-1047 or UTF-8 JSON data

i.e.
**HWICMD**
- HWI_CMD_OSCMD

**Input Format (Parameters):**

```
myCmdParm.PriorityType = Hwi_CMD_Priority
myCmdParm.OSCMDString = 'd a,l'
```

i.e.
**HWIREST**
- POST /api/logical-partitions/{object-id}/operations/send-os-cmd

**Input Format (Request Body):**

```
{
    "is-priority": true,
    "operating-system-command-text": "d a,l"
}
```

**HWIQUERY**
- myQueryParm.1.ATTRIBUTEIDENTIFIER = HWI_NAME

**Output Format (Parameters):**

```
Say myQueryParm.1.ATTRIBUTEVALUE
'LPAR1'
```

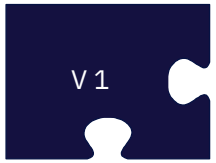
**HWIREST**
- GET /api/logical-partitions/{object-id}?properties=name

**Output Format (Response Body):**

```
{
    "name": "LPAR1"
}
```

We highly recommend using the **z/OS Client Web Enablement Toolkit JSON Parser**, which is built into the z/OS base, for all your JSON parsing needs

# Usage & Invocation – v1 vs v2 Error Information



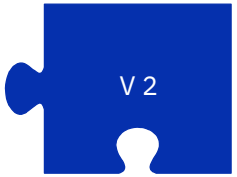

- ReturnCode
- DiagArea.

i.e.
**HWICONN**
- ReturnCode
- DiagArea.
- myInConnectToken
- myOutConnectToken
- myConnectType
- myConnectTypeValue

**Response Error Information**
- **ReturnCode**
- **DiagArea.**

```
Say 'API ReturnCode = 'ReturnCode'.'
If (ReturnCode<>0) Then
    Do
        Say ' Diag_index=' DiagArea.DIAG_INDEX
        Say ' Diag_key=' DiagArea.DIAG_KEY
        Say ' Diag_actual=' DiagArea.DIAG_ACTUAL
        Say ' Diag_expected=' DiagArea.DIAG_EXPECTED
        Say ' Diag_commerr=' DiagArea.DIAG_COMMERR
        Say ' Diag_text=' DiagArea.DIAG_TEXT
    End
End
```



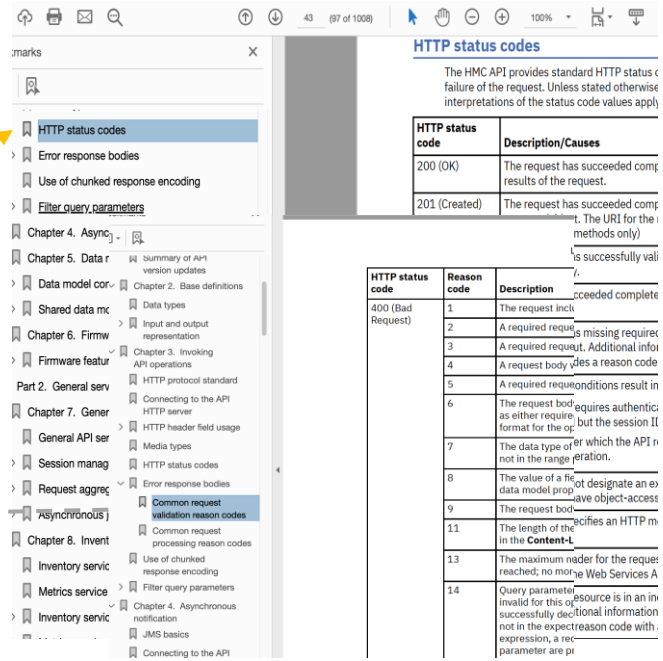


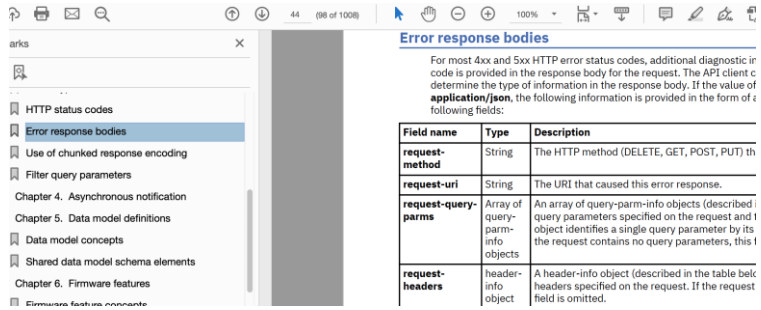

- HttpStatus
- ReasonCode
- ResponseBody

i.e.
**HWIREST**
- POST /api/logical-partitions/{object-id}/operations/send-os-cmd

**Response Error Information**
- **HttpStatus: 400 (Bad Request)**
- **ReasonCode: 6**
- **Response Body:**


```
{
    "http-status": 400,
    "reason": 6,
    "request-uri": "/api/cpcs/aaa97f93-3072-310c-a554-e5c26b1655dd",
    "request-method": "POST",
    "message": "The JSON object contains an unrecognized field ('fred')",
    "request-headers": {
        "authorization": "Basic YmNwaWlwOmJjcGlpcA==",
        "content-length": "19",
        "postman-token": "8972dc34-be8f-44cb-9614-7a7478c11375",
        "host": "9.12.16.166:6794",
        "content-type": "application/json",
        "connection": "keep-alive",
        "x-api-session": "5wd7sgmgsui9j4a5h0asoty781hqgaqttikk8g1j5s93pstp77",
        "cache-control": "no-cache",
        "accept-encoding": "gzip, deflate, br",
        "user-agent": "PostmanRuntime/7.24.1",
        "accept": "*/*"
    },
    "request-body-as-string": "{\n  \"fred\":false\n}\n",
    "request-body-as-string-partial": false,
    "request-authenticated-as": "BCPiiP"
}
```

# Usage & Invocation - REXX Syntax

V 2

**HWIREST - REXX interface**

address bcpii “hwirest requestParm. responseParm.”

Where requestParm and responseParm stem contains compound (stem) variables which represent input and output parameters for the requested command

**requestParm details**

requestParm.httpMethod
requestParm.uri
requestParm.targetName
requestParm.requestBody
requestParm.clientCorrelator
requestParm.encoding
requestParm.requestTimeout

**responseParm details**

responseParm.responseDate
responseParm.requestId
responseParm.location
responseParm.responseBody
responseParm.httpStatus
responseParm.reasonCode

# Usage & Invocation – C/ASM Syntax

V 2

**Non-REXX parameters**

HWIREST( requestParmPtr, responseParmPtr)

where requestParmPtr is of **REQUEST_PARM_TYPE** type

where responseParmPtr is of **RESPONSE_PARM_TYPE** type

**REQUEST_PARM_TYPE** sample definition in C/C++ IDF:

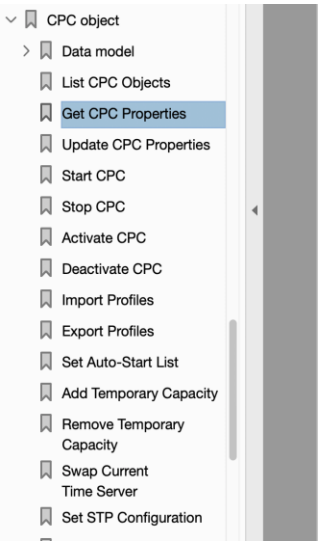
```
typedef struct [
      HTTPMETHOD_TYPE httpMethod;
      const char     *uri;
      unsigned int    uriLen;
      const char     *targetName;
      unsigned int    targetNameLen;
      const char     *requestBody;
      unsigned int    requestBodyLen;
      const char     *clientCorrelator;
      unsigned int    clientCorrelatorLen;
      ENCODING_TYPE   encoding;
      unsigned int    requestTimeout;
 ] REQUEST_PARM_TYPE;
```

**RESPONE_PARM_TYPE** sample definition in C/C++ IDF:

```
typedef struct [
      char           *responseDate;
      unsigned int    responseDateLen;
      char           *requestId;
      unsigned int    requestIdLen;
      char           *location;
      unsigned int    locationLen;
      char           *responseBody;
      unsigned int    responseBodyLen;
      unsigned int    httpStatus;
      int             reasonCode;
 ] RESPONSE_PARM_TYPE;
```

# Usage & Invocation – request parameter

V 2




## Get CPC Properties

The `Get CPC Properties` operation retrieves the properties of a single CPC object designated by *{cpc-id}*.

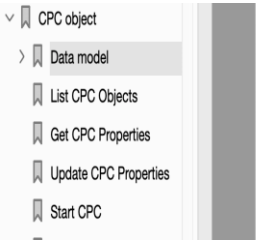
### HTTP method and URI

```
GET /api/cpcs/{cpc-id}
```

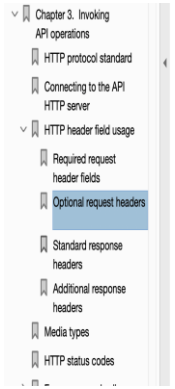
In this request, the URI variable *{cpc-id}* is the object ID of the target CPC object.

**Query parameters:**

| Name | Type | Rqd/Opt | Description |
|---|---|---|---|
| properties | List of String Enum | Optional | Filter string to limit returned properties to those that are identified here. This is a list of comma-separated strings where each string is a property name defined in the CPC object's data model. |
| cached-acceptable | Boolean | Optional | Indicates whether cached values are acceptable for the returned properties. Valid values are **true** and **false**. The default is **false**. |




| | | | |
|---|---|---|---|
| allowed | | | |
| ec-mcl-description | – | ec-mcl-description object | Describes the Engineering Change (EC) and MicroCode Level (MCL) for the CPC object. An empty object is returned if the information is unavailable from the SE. Refer to the description of the `ec-mcl-description` object for details. |
| has-automatic-se-switch-enabled | – | Boolean | Automatic switching between primary and alternate Support Elements is enabled for the CPC object (true), or |

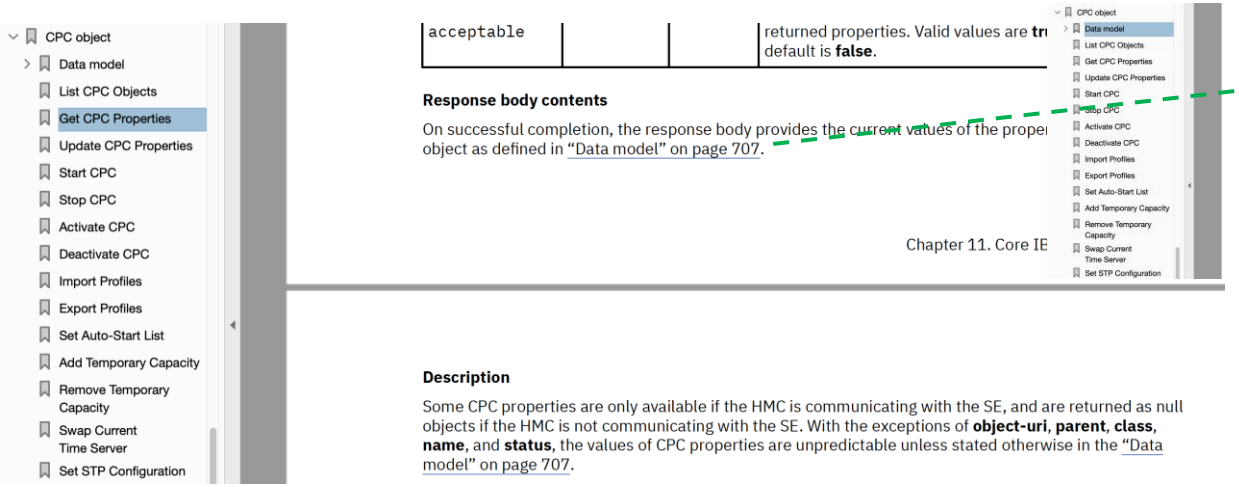


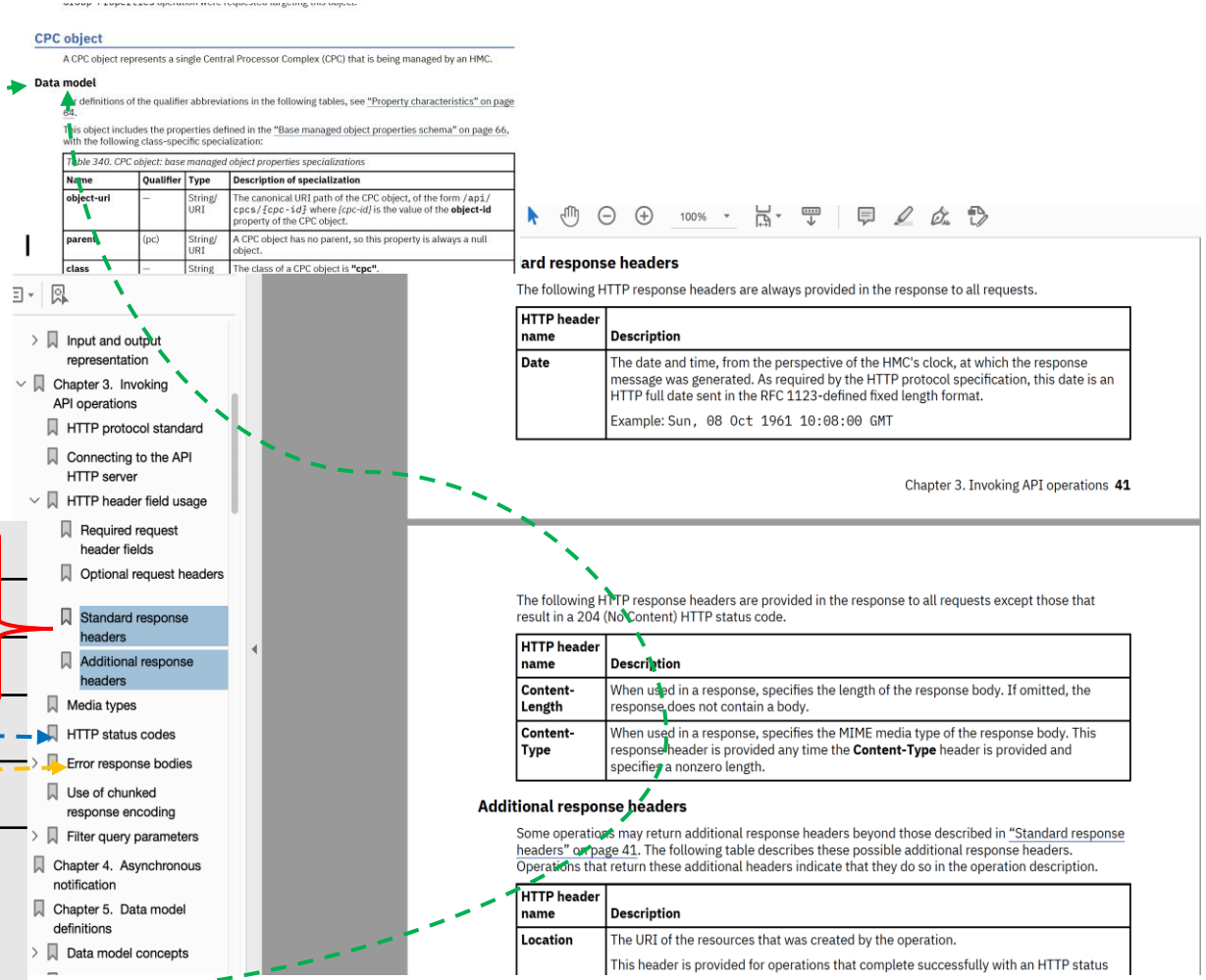

| | | |
|---|---|---|
| | | API user's HMC login i improved audit logging behalf of multiple upst single HMC login identi their upstream user in upstream users can be use up to the first 64 c present, and silently ig longer than 64 charact |
| X-Client-Correlator | Optional | A string that provides that is of significance t the like. The HMC will trace or log data it coll information with simila supplied in this header determination and doe The HMC will use up to header if present, and if it is longer than 64 c |

| | |
|---|---|
| requestParm.httpMethod | **HWI_REST_GET** |
| requestParm.uri | /api/cpcs/0583cc7f-5b24-3400-a7da-d30e14233684?properties=ec-mcl-description |
| requestParm.targetName | IBM390PS.T115 (netid.nau associated with the CPC) |
| requestParm.requestBody | n/a (Get CPC Properties does not accept a request body) |
| requestParm.clientCorrelator | IntegrationTeam |
| requestParm.encoding | 0 (default to IBM-1047, supports IBM-1047 and UTF-8) |
| requestParm.requestTimeout | 0 (in ms, defaults to 60 minute) |

# Usage & Invocation – response parameter

V 2




| | |
|---|---|
| responseParm.responseDate | Wed, 06 May 2020 18:05:36 GMT |
| responseParm.location | n/a (only returned when a new resource is created) |
| responseParm.requestId | Sxb939b63c-7455-11ea-9f91-00106f23d4ce.1f3 Rxc |
| responseParm.httpStatus | 200 (OK) |
| responseParm.reasonCode | 0 |
| responseParm.responseBody | {<br>"parent": null,<br>"processor-count-pending-general-purpose": 0,<br>"cbu-number-of-tests-left": 0,<br>"msu-permanent-plus-temporary": 7696,<br>"processor-count-pending": 0,<br>"acceptable-status": [<br>"operating"<br>],<br>"processor-count-defective": 0,<br>"network1-ipv4-mask": "255.255.255.0",<br>"machine-serial-number": "0000200DBB57",<br>"processor-running-time-type": "system-determined",<br>"zcpc-maximum-potential-power": 27558,<br>"storage-customer": 2949120,<br>... <trimmed>...<br>} |

# Usage & Invocation – SAF Security


- BCPii security relies on SAF authorization (specific profiles defined in the FACILITY class)

- In BCPii V1 APIs, BCPii can automatically build the profile name to check against SAF, based on either what the user was connecting to or the connect token passed in

- In BCPii V2 API
  - no connect token available for BCPii to build the profile name
  - naming convention of the entity being referenced in the URI is not in the same format as the SAF profile
  - targetName input parameter will be used to help BCPii build the profile name
    - Each entity returned in the response from a list URI request will have an associated target-name property
      - Application will simply use this returned value on all subsequent calls associated with that entity
    - The SE will also validate to make sure the targetName matches the entity referenced in the URI

# Usage & Invocation – SAF Security

Naming convention format V1 vs V2

| Entity | Referenced as (V 1) | Referenced as (V 2) | Profile Name (SAF) |
|---|---|---|---|
| CPC<br>i.e. name: T115 | netid.nau<br>i.e. IBM390PS.T115 | Object ID<br>/api/cpcs/*{cpc-id}*<br>i.e.<br>• *{cpc-id} : aaa97f93-3072-310c-a554-e5c26b1655dd*<br>• target-name: IBM390PS.T115 | HWI.**TARGET**.netid.nau |
| Image<br>i.e. name: BCPE | netid.nau.imagename<br>i.e. IBM390PS.T115.BCPE | Object ID<br>/api/logical-partitions/{logical-partition-id*}*<br>i.e.<br>• *{*logical-partition-id*} : aaa97f93-3072-310c-a554-e5c26b1655dd*<br>• target-name: IBM390PS.T115.BCPE | HWI.**TARGET**.netid.nau.imagename |
| Capacity Record<br>i.e. name: MARIANNE | netid.nau.caprec<br>i.e. IBM390PSE.T115.MARIANNE | Object ID<br>/api/cpcs/*{cpc-id}*/capacity-records/*MARIANNE*<br>i.e.<br>• *{capacity-record-id} : MARIANNE*<br>• target-name: IBM390PS.T115. MARIANNE | HWI.**CAPREC**.netid.nau.caprec |
| JOB Uri | netid.nau<br>netid.nau.imagename | *The 'target-name' used on the asynchronous REST API that returned the JOB Uri (Cap Rec URIs are all synchronous)* | HWI.**TARGET**.netid.nau<br>HWI.**TARGET**.netid.nau.imagename |

# Usage & Invocation – Invocation Example

V 2

```
/*
  A previous HWIREST request performed a LIST on CPCs using GET /api/cpcs  and stored the uri and target name associated with CPC R32:
             CPCR32 = "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340”
             CPCR32TargetName = “IBM390PS.R32”

  To retrieve specific properties associated with CPC R32, we use the REST API for Get CPC Properties: GET /api/cpcs/{cpc-id}
  taking advantage of the optional properties Query Parameter to scope the request to only retrieve what we want

  NOTE: the following HWIREST request for CPC R32 requires READ authorization to SAF PROFILE HWI.TARGET.IBM390PS.R32
*/

requestParm.httpMethod = HWI_REST_GET
requestParm.uri = CPCR32||’?properties=storage-total-installed&cached-acceptable=true’
requestParm.targetName = CPCR32TargetName

address bcpii "HWIREST" “requestParm.” “responseParm.”

If responseParm.httpStatus > 199 & responseParm.httpStatus < 300 Then
  Do
    Say ‘request finished successfully with HTTPStatus’|| responseParm.httpStatus
    Parse responseParm.responseBody and retrieve either the storage-total-installed value
  End
Else
  Do
    Say ‘request failed with HTTPStatus’|| responseParm.httpStatus
    Say ‘and reasonCode is’|| responseParm.reasonCode

    If LENGTH(responseParm.responseBody) > 0 then
      Parse responseParm.responseBody and retrieve either the error information
  End
```

# Usage & Invocation – HWILIST vs HWIREST

V 1

**HWILIST**

- HWI_LIST_CPCS

Returned:

*Blank-delimited list of CPC names*

```
M44 T115
```

V 2

**HWIREST**

- GET /api/cpcs

Returned:


```
{
    "cpcs": [
      {
          "name": "M44",
          "se-version": "2.14.1",
          "object-uri": "/api/cpcs/ab494a2f-c28e-3909-9dab-c57996d25bdd",
          “location”: “remote”,
          "target-name”: “IBM390PS.M44”
      },
      {
          "name": "T115",
          "se-version": "2.15.0",
          "object-uri": "/api/cpcs/0583cc7f-5b24-3400-a7da-d30e14233684",
          “location”: “local”,
          "target-name”: “IBM390PS.T115”
      }
      ]
}
```

# Usage & Invocation – HWICONN vs HWIREST

V 1

**HWICONN**

- HWI_CPC
- ‘IBM390PS.M44’

Returned: CPCConnectToken

V 2

**HWIREST**

- GET /api/cpcs

Returned:


```
{
    "cpcs": [
      {
          "name": "M44",
          "se-version": "2.14.1",
          "object-uri": "/api/cpcs/ab494a2f-c28e-3909-9dab-c57996d25bdd",
          “location”: “remote”,
          "target-name”: “IBM390PS.M44”
       },
      {
          "name": "T115",
          "se-version": "2.15.0",
          "object-uri": "/api/cpcs/0583cc7f-5b24-3400-a7da-d30e14233684",
          “location”: “local”,
          "target-name”: “IBM390PS.T115”
        }
       ]
}
```

# Usage & Invocation – Local Entity

V 1

**HWICONN**

allows the user to specify '*' to indicate the local CPC or Image

i.e.

- HWI_CPC
- '*'

Returned: CPCConnectToken

V 2

**HWIREST**

- CPC data model will include a **location** string property that indicates if the CPC is **local** or **remote**
- LPAR data model will include a **request-origin** Boolean property that indicates if the LPAR is local **(true)** or remote **(false**

i.e.

- `GET /api/cpcs`

Returned:

```
{
  "cpcs": [
      {
          "name": "M44",
          "se-version": "2.14.1",
          "object-uri": "/api/cpcs/ab494a2f-c28e-3909-9dab-c57996d25bdd",
          "location": "remote",
          "target-name": "IBM390PS.M44"
       },
      {
          "name": "T115",
          "se-version": "2.15.0",
          "object-uri": "/api/cpcs/0583cc7f-5b24-3400-a7da-d30e14233684",
          "location": "local",
          "target-name": "IBM390PS.T115"
       }
       ]
}
```

# Usage & Invocation – HWIQUERY vs HWIREST

V 1

**HWIQUERY**

- List of 1 to 64 attributes explicitly specified

- Specified:

```
myQueryParm.0 = 4 /* Set number of attributes */
myQueryParm.1.ATTRIBUTEIDENTIFIER = HWI_NAME
myQueryParm.2.ATTRIBUTEIDENTIFIER = HWI_LUAPROF
myQueryParm.3.ATTRIBUTEIDENTIFIER = HWI_MSERIAL
myQueryParm.4.ATTRIBUTEIDENTIFIER = HWI_IPADDR
```

```
address bcpii “hwiquery RetCode myConnectToken
myQueryParm. myDiag.”
```

Returned:

```
If rc = 0 & Retcode = 0 Then
  Do n=1 to myQueryParm.0
    Say “Parameter”||n||“:”myQueryParm.n.ATTRIBUTEVALUE
  End
```

V 2

**HWIREST**

**GET /api/cpcs/{cpc-id}** or **GET /api/cpcs/{cpc-id}?properties=x,y,z**

- For selected data models (e.g., CPCs, LPARs, Caprecs), there is a choice to explicitly specify the attributes (properties) (See Query parms for details)
- **cached-acceptable** – use if possible, for significantly better performance
- Specified:

```
CPCObjectURI is the value associated with the object-uri property of the CPC returned from the
          GET /api/cpcs request
TargetName is the value associated with the target-name property of the CPC returned from the
          GET /api/cpcs request
```

```
QueryParms = ‘?properties=name,last-used-activation-profile,machine-serial-number,
                network1-ipv4-pri-ipaddr&cached-acceptable=true’
userRequest.httpmethod = HWI_REST_GET
userRequest.uri = CPCObjectURI||QueryParms
userRequest.targetname = TargetName
```

```
address bcpii “HWIREST userRequest. response.”
```

Returned:

```
If rc = 0 & response.httpStatus = 200 Then
   Success
```

```
If response.responseBody <> ‘’ Then
  Parse JSON returned in response.responseBody
```

# Usage & Invocation – HWISET vs HWIREST

V 1

## HWISET, HWISET2

- Set 1 to 9 attributes specified
  Specify connect token, attribute and value for each attribute

Specified:

```
mySetParm.0 = 2 /* Set number of attributes */

mySetParm.1.SET2_CTOKEN = CPCConnectToken
mySetParm.1.SET2_SETTYPE = HWI_ACCSTAT
mySetParm.1.SET2_SETVALUE = HWMCA_STATUS_OPERATING

mySetParm.2.SET2_CTOKEN = CPCConnectToken
mySetParm.2.SET2_SETTYPE = HWI_PRUNTYPE
mySetParm.2.SET2_SETVALUE = HWMCA_DETERMINED_SYSTEM
```

```
address bcpii “hwiset RetCode CPCConnectToken mySetParm.
myDiag.”
```

Returned:

```
If rc = 0 & Retcode = 0 Then
  Success
```

V 2

## HWIREST

**POST /api/cpcs/{cpc-id}**

- Explicitly specify the attributes (properties) that you want to modify
- Consult data model to learn which attributes are updatable
- Selected capacity attributes can be set for different LPARs on the same CPC via the **POST /api/ cpcs/{cpc-id}/operations/update-lpar-controls** URI

Specified:

```
CPCObjectURI is the value associated with the object-uri property of the CPC returned from the
          GET /api/cpcs request
TargetName is the value associated with the target-name property of the CPC returned from the
          GET /api/cpcs request
```

```
userRequest.httpmethod = HWI_REST_POST
userRequest.uri = CPCObjectURI
userRequest.targetname = TargetName
userRequest.requestbody = ‘{"acceptable-status" : ["operation"],
                            "processor-running-time-type" : "system-determined"}’
address bcpii “HWIREST userRequest. response.”
```

Returned:

```
If rc = 0 & response.httpStatus = 204 Then
   Success
```

# Usage & Invocation – HWICMD vs HWIREST

V 1

## HWICMD

- Fill in unique parameters for the command
  - Assembler DSECT
  - C structure
  - REXX stem variables
- Pass in CPC or image connect token
- Call synchronous command

Specified:

```
CmdType = HWI_CMD_LOAD
myCmdParm.LoadAddr = ‘0980’
myCmdParm.LoadParm = ‘0224MDX’
myCmdParm.ForceType = HWI_CMD_FORCE
```

```
address bcpii “hwicmd RetCode myConnectToken
CmdType myCmdParm. myDiag.”
```

Returned:

```
If rc = 0 & Retcode = 0 Then
  Command was accepted by the SE
```

Listen for ENF68 Command completion event to learn of final response. Requires:

1. An HWIEVENT call to register your ENF exit and at a minimum, listen for Hwi_Event_CmdResp PRIOR TO INVOKING COMMAND
2. Coding an ENF exit in Metal C or Assembler
3. If REXX, requires a “helper” application that can wait on an ECB, which the ENF exit will post when Command response has been received

V 2

## HWIREST

**POST /api/cpcs/{cpc-id}/logical-partitions**

- Same restrictions apply to issuing “commands” from ISV or TSO REXX

Specified:

```
LPARObjectURI is the value associated with the object-uri property of the LPAR returned
from the GET /api/cpcs/{cpc-id}/logical-partitions request
```

```
LPARTargetName is the value associated with the target-name property of the LPAR returned
from the GET /api/cpcs/{cpc-id}/logical-partitions request
```

```
userRequest.httpmethod = HWI_REST_POST
userRequest.uri = LPARObjectURI||‘/operations/load’
userRequest.targetname = LPARTargetName
userRequest.requestBody = ‘{
                              "load-address":"0980",
                              "load-parameter":"0224MDX"
                              "clear-indicator": true,
                              "timeout" : 60,
                              "store-status-indicator": true,
                              "force":true}’
```

```
address bcpii “HWIREST userRequest. response.”
```

Returned:

```
If rc = 0 & response.httpStatus = 200 Then
   Command was accepted by the SE
```

1. Parse JSON returned in response.responseBody for the **job-uri** for this command.
2. Using HWIREST, poll SE for results of operation

# Usage & Invocation – HWICMD vs HWIREST

- Many HWICMD/HWICMD2 requests have equivalent HWIREST POST /operations URI requests; a few use a GET URI
  - for example:
    - HWI_CMD_LOAD -> POST /api/logical-partitions/{logical-partition-id}/operations/load
    - HWI_CMD_ACTIVATE -> POST /api/logical-partitions/{logical-partition-id}/operations/activate
    - HWI_CMD_HWMSG -> GET /api/cpcs/{cpc-id}/hardware-messages

- Like HWICMD/HWICMD2, a *similar* TSO/E REXX and ISV REXX restriction for command equivalent requests exists for HWIREST.
  - Applications executing from either a TSO/E REXX or an ISV REXX environment are restricted from issuing POST and DELETE '/operations/' requests that includes '/operations/' in the URI
  - One exception is: POST /api/cpcs/{cpc-id}/operations/update-lpar-controls

    This URI allows LPAR weight query/updates across an entire CPC and is equivalent to issuing HWISET2

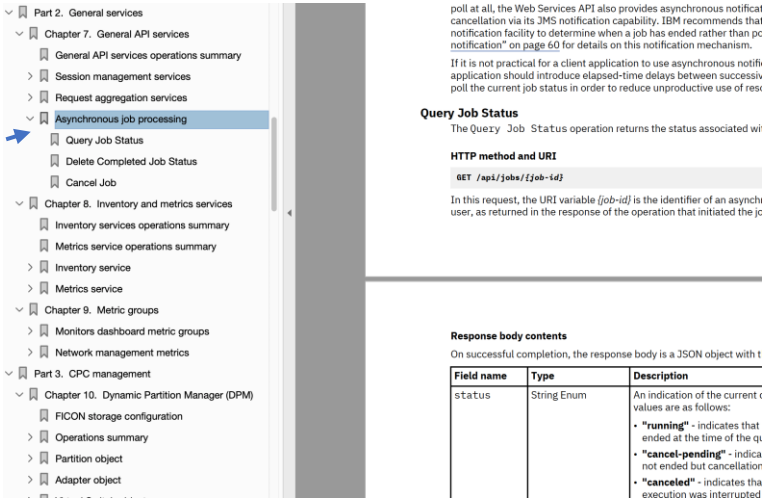
# Usage & Invocation - Events

V 1

- HWIEVENT to register/deregister to event notification
  - PUSH methodology
- BCPii signals the appropriate ENF68 event
- Ability to listen for all events (even ones don't originate)

V 2

- Asynchronous job processing
  - PULL methodology, akin to BCPii event notification in USS (HwiGetEvent)
  - Originating request will return a response body that will contain a job-uri that can be used to query the outcome
  - Specific to events you originate or know the job-uri



# Usage & Invocation – HWIEVENT

- Except for registering for HWIREST command completions, HWIEVENT usage value remains the same
- Requires:
  - Writing and deploying an ENF 68 exit written in Assembler or Metal C
  - Invoking HWIEVENT or the BCPii z/OS UNIX services to register for one or more unsolicited events, pointing to the ENF 68 exit deployed
  - ENF68 exit  receives push notifications from BCPii when the event has taken place
  - Code likely wakes up a waiting program to perform processing outside the ENF 68 exit as a result of the event

# Usage & Invocation – HWIREST SMF 106

BCPii will continue to generate SMF 106 records for the successful updating of hardware resources

- **SMF 106 header/self-defining section**
  - **V2: new fields added** (backward compatible with existing BCPii)
- **SMF 106 Subtype 1 generation**
  - **V1:** HWISET/HWISET2 APIs that complete with a zero return code
  - **V2:** HWIREST API calls that issued a POST/PUT/DELETE request for URI that does NOT contain 'operations' that complete with an HTTP status in the 200 range
- **SMF 106 Subtype 2 generation**
  - **V1:** HWICMD/HWICMD2 APIs that complete with a zero return code (accepted by the SE)
  - **V2:** HWIREST API calls that issued a POST/PUT/DELETE request for URI that contains 'operations' that complete with an HTTP Status in the 200 range

**ICETOOL has been updated to provide a simple non-programming sample of extracting SMF 106 data for V2**

# Usage & Invocation – Comparison – retrieve Image GPP weight

Retrieval of image/LPAR information

- What is the initial general purpose processor processing weight
- Is the initial general purpose processor processing weight capped?

The CPC and Image(LPAR) names are known

- CPC is 'IBM390PS.T115'
- Image is 'BCPE'

# Usage & Invocation – Comparison – retrieve Image GPP weight

| | Step 1 | Step 2 | Step 3 | Step 4 |
|---|---|---|---|---|
| v1 | **HWICONN**<br>• HWI_CPC<br>• 'IBM390PS.T115'<br><br>Returned: CPCConnectToken | **HWICONN**<br>• CPCConnectToken<br>• HWI_IMAGE<br>• BCPE<br>Returned: ImageConnectToken | **HWIQUERY**<br>• HWI_SGPIPW<br>• HWI_SGPIPWCAP<br>Returned:<br>*HWI_SGPIPW=10*<br>*HWI_SGPIPWCAP=0* | **HWIDISC**<br>• ImageConnectToken<br>Followed by….<br>• CPCConnectToken |
| v2 | **HWIREST**<br>• GET /api/cpcs?name=T115<br>Returned:<br>*{*<br>*"cpcs": [{*<br>*"name": "T115",*<br>*"se-version": "2.15.0",*<br>*"object-uri": "/api/cpcs/0583cc7f-5b24-3400-a7da-d30e14233684",*<br>*"location": "local",*<br>*"target-name": "IBM390PS.T115"*<br>*}]*<br>*}* | **HWIREST**<br>• GET /api/cpcs/0583cc7f-5b24-3400-a7da-d30e14233684/logical-partitions?name=BCPE<br>• targetName = "IBM390PS.T115"<br><br>Returned:<br>*{*<br>*"logical-partitions": [{*<br>*"name": "BCPE",*<br>*"object-uri": "/api/logical-partitions/b45fd9d0-d368-3557-8038-007f91ce6b19",*<br>*"status": "operating",*<br>*"request-origin" : true,*<br>*"target-name": "IBM390PS.T115.BCPE"*<br>*}]*<br>*}* | **HWIREST**<br>• GET /api/logical-partitions/b45fd9d0-d368-3557-8038-007f91ce6b19?properties=initial-processing-weight,initial-processing-weight-capped&cached-acceptable=true<br>• targetName = "IBM390PS.T115.BCPE"<br><br>Returned:<br>*{*<br>*"initial-processing-weight-capped": false,*<br>*"initial-processing-weight": 10*<br>*}* | |

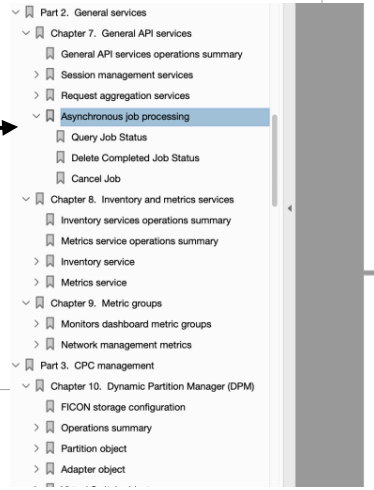
# Usage & Invocation – Comparison – re-activate an LPAR

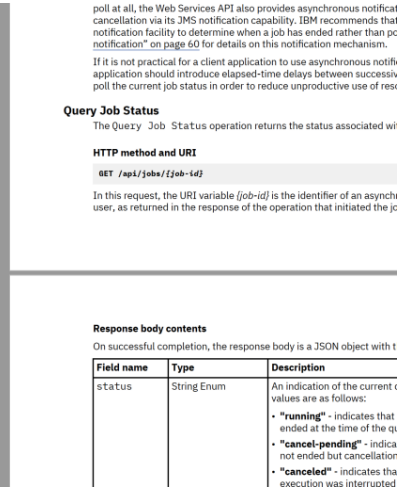
In the following example, the Image/LPAR information is already known

- V1: A connection token has already been obtained for the Image
- V2: The image URI and corresponding target name have been obtained

# Usage & Invocation – Comparison – re-activate an LPAR

| | Step 1 | Step 2 | Step 3 |
|---|---|---|---|
| v1 | **HWIEVENT**<br>• *Hwi_Event_CmdResp* | **HWICMD**<br>• *HWI_CMD_ACTIVATE* | Listen for **ENF 68** broadcast |
| v2 | | **HWIREST**<br>• POST /api/logical-partitions/b45fd9d0-d368-3557-8038-007f91ce6b19/operations/activate<br>• targetName = “IBM390PS.T115.BCPE"<br><br>Returned:<br>{<br>"job-uri":"/api/jobs/3ae9a3c8-55b3-11e7-b883-fa163e6e13bb"<br>} | *... Repeat the following until “status” indicates job completion*<br>**HWIREST**<br>• GET /api/jobs/3ae9a3c8-55b3-11e7-b883-fa163e6e13bb<br>• targetName = “IBM390PS.T115.BCPE"<br><br>Returned:<br>{<br>"job-reason-code": 0,<br>"job-status-code": 200,<br>"status": "complete"<br>} |



# Interactions & Dependencies

- Software Dependencies
  - None

- **Hardware Dependencies**
  - Limited to z15 and later systems
    - SE 2.15.0 - MCL **P46598.370**, Bundle **S38**
    - HMC 2.15.0 - MCL **P46686.001**, Bundle **H25**

- Exploiters
  - None

# Upgrade & Coexistence Considerations

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

- **Coexistence (cross system) requirement**
  - Firmware update required on:
    - Local CPC where the BCPii HWIREST service is used (SE)
    - HMC associated with local CPC
    - Any other CPC that is targeted by a BCPii HWIREST request (SE and perhaps other HMC(s) )

# Installation & Configuration

- This functionality is rolled down to z/OS 2.4 via APAR OA60351
  - IPL is required after applying the APAR

# Summary

- z/OS BCPii is introducing a new API: **HWIREST**
- With the introduction of HWIREST, applications can instantly access numerous additional attributes, including CPC storage, storage allocated for LPARs, CPC environmentals, more detailed information about processors and their assignments to an LPAR, and countless other properties. Numerous future attributes introduced with a new HW Level will also be automatically available using this API, without any additional z/OS update.

# Appendix

- Publication References
  - IBM Z/OS MVS Programming: Callable Services for High-Level Languages
  - **Hardware Management Console Web Services API**
    - available on Resource Link: http://www.ibm.com/servers/resourcelink
      - Library → z15 → Web Services API
  - MVS System Management Facilities (SMF)
    - BCPii SMF 106
  - MVS System Codes
    - BCPii System Code 42

- sample will be available on Github: https://github.com/IBM/zOS-BCPii