

# **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

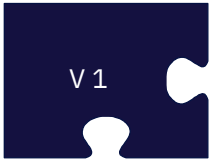


# Usage & Invocation – v1 vs v2

---

## 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



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

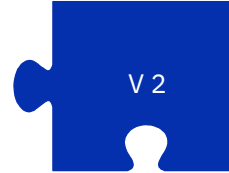
i.e.

## HWICMD

- HWI\_CMD\_OSCMD

### Input Format (Parameters):

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



IBM-1047 or UTF-8 JSON data

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.



- HttpStatus
- ReasonCode
- ResponseBody

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\_commerk=' DiagArea.DIAG\_COMMERR

Say ' Diag\_text=' DiagArea.DIAG\_TEXT

End

End

© 2021 IBM Corporation

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": "5wd7sgmgsui9j4a5h0asoty781hqqagttikk8g1j5s93pstp77",
    "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"
}
```

marks

HTTP status codes

Error response bodies

Use of chunked response encoding

Filter query parameters

Chapter 4. Asynchronous

Chapter 5. Data model

Shared data model

Chapter 6. Firmware

Part 2. General services

Chapter 7. General API services

Session management

Request aggregation

Asynchronous

Chapter 8. Inventory services

Inventory services

Metrics services

Inventory services

HTTP status codes

The HMC API provides standard HTTP status codes for the request. Unless stated otherwise, the status code values apply to the request.

HTTP status code	Description/Causes
200 (OK)	The request has succeeded completely.
201 (Created)	The request has succeeded completely. The URI for the new resource is provided in the response body.
400 (Bad Request)	The request is malformed. The following table lists the reasons for the error.

HTTP status code	Reason code	Description
400 (Bad Request)	1	The request is malformed.
	2	A required request header is missing.
	3	A required request body is missing.
	4	A request body contains a reason code.
	5	A required request condition is not met.
	6	The request body requires authentication as either a session ID or a session ID and a session ID.
	7	The data type of the request body is not in the range of the expected data type.
	8	The value of a field in the request body does not designate an expected object-access.
	9	The request body contains an invalid HTTP method.
	11	The length of the request body is not in the range of the expected length.
	13	The maximum number of request parameters is exceeded; no more Web Services API parameters are allowed.
	14	Query parameters are invalid for this resource. In an invalid query parameter, the resource is in an invalid state.

marks

HTTP status codes

Error response bodies

Use of chunked response encoding

Filter query parameters

Chapter 4. Asynchronous

Chapter 5. Data model definitions

Data model concepts

Shared data model schema elements

Chapter 6. Firmware features

Firmware features concepts

Error response bodies

For most 4xx and 5xx HTTP error status codes, additional diagnostic information is provided in the response body for the request. The API client can determine the type of information in the response body. If the value of application/json, the following information is provided in the form of a JSON object:

Field name	Type	Description
request-method	String	The HTTP method (DELETE, GET, POST, PUT) that caused this error response.
request-uri	String	The URI that caused this error response.
request-query-params	Array of query-param-info objects	An array of query-param-info objects (described below) that identifies a single query parameter by its name and value.
request-headers	header-info object	A header-info object (described in the table below) that identifies a single header specified on the request. If the request field is omitted, the header is not present.

# Usage & Invocation - REXX Syntax



## 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



## 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



- ▼ CPC object
  - > Data model
    - List CPC Objects
    - Get CPC Properties
    - Update CPC Properties
    - Start CPC
    - Stop CPC
    - Activate CPC
    - Deactivate CPC
    - Import Profiles
    - Export Profiles
    - Set Auto-Start List
    - Add Temporary Capacity
    - Remove Temporary Capacity
    - Swap Current Time Server
    - Set STP Configuration

- ▼ Chapter 3. Invoking API operations
  - HTTP protocol standard
  - Connecting to the API HTTP server
  - ▼ HTTP header field usage
    - Required request header fields
    - Optional request headers
    - Standard response headers
    - Additional response headers
    - Media types
    - HTTP status codes

## 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 <b>true</b> and <b>false</b> . The default is <b>false</b> .

- ▼ CPC object
  - > Data model
    - List CPC Objects
    - Get CPC Properties
    - Update CPC Properties
    - Start CPC

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-switch-enabled	-	Boolean	Automatic switching between primary and alternate Support Elements is enabled for the CPC object (true) or

		INCLUDED BY API USER API user's HMC login id (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 c that is of significance b the like. The HMC will i trace or log data n co 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 cl

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

# Usage & Invocation – response parameter



CPC object

Data model

List CPC Objects

Get CPC Properties

Update CPC Properties

Start CPC

Stop CPC

Activate CPC

Deactivate CPC

Import Profiles

Export Profiles

Set Auto-Start List

Add Temporary Capacity

Remove Temporary Capacity

Swap Current Time Server

Set STP Configuration

acceptable		returned properties. Valid values are true or false. Default is false.
------------	--	--

**Response body contents**

On successful completion, the response body provides the current values of the properties of the CPC object as defined in "Data model" on page 707.

**Description**

Some CPC properties are only available if the HMC is communicating with the SE, and are returned as null objects if the HMC is not communicating with the SE. With the exceptions of **object-uri**, **parent**, **class**, **name**, and **status**, the values of CPC properties are unpredictable unless stated otherwise in the "Data model" on page 707.

CPC object

Data model

List CPC Objects

Get CPC Properties

Update CPC Properties

Start CPC

Stop CPC

Activate CPC

Deactivate CPC

Import Profiles

Export Profiles

Set Auto-Start List

Add Temporary Capacity

Remove Temporary Capacity

Swap Current Time Server

Set STP Configuration

Chapter 11. Core IE

CPC object

Data model

List CPC Objects

Get CPC Properties

Update CPC Properties

Start CPC

Stop CPC

Activate CPC

Deactivate CPC

Import Profiles

Export Profiles

Set Auto-Start List

Add Temporary Capacity

Remove Temporary Capacity

Swap Current Time Server

Set STP Configuration

Chapter 3. Invoking API operations

Input and output representation

Chapter 3. Invoking API operations

HTTP protocol standard

Connecting to the API HTTP server

HTTP header field usage

Required request header fields

Optional request headers

Standard response headers

Additional response headers

Media types

HTTP status codes

Error response bodies

Use of chunked response encoding

Filter query parameters

Chapter 4. Asynchronous notification

Chapter 5. Data model definitions

Data model concepts

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

ard response headers

The following HTTP response headers are always provided in the response to all requests.

HTTP header name	Description
Date	The date and time, from the perspective of the HMC's clock, at which the response message was generated. As required by the HTTP protocol specification, this date is an HTTP full date sent in the RFC 1123-defined fixed length format. Example: Sun, 08 Oct 1961 10:08:00 GMT

Chapter 3. Invoking API operations 41

The following HTTP response headers are provided in the response to all requests except those that result in a 204 (No Content) HTTP status code.

HTTP header name	Description
Content-Length	When used in a response, specifies the length of the response body. If omitted, the response does not contain a body.
Content-Type	When used in a response, specifies the MIME media type of the response body. This response header is provided any time the Content-Type header is provided and specifies a nonzero length.

Additional response headers

Some operations may return additional response headers beyond those described in "Standard response headers" on page 41. The following table describes these possible additional response headers. Operations that return these additional headers indicate that they do so in the operation description.

HTTP header name	Description
Location	The URI of the resources that was created by the operation. This header is provided for operations that complete successfully with an HTTP status

# 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 	Referenced as 	Profile Name 
CPC i.e. name: T115	netid.nau i.e. IBM390PS.T115	Object ID /api/cpcs/{cpc-id} i.e. <ul style="list-style-type: none"><li>• {cpc-id} : aaa97f93-3072-310c-a554-e5c26b1655dd</li><li>• target-name: IBM390PS.T115</li></ul>	HWI. <b>TARGET</b> .netid.nau
Image i.e. name: BCPE	netid.nau.imagename i.e. IBM390PS.T115.BCPE	Object ID /api/logical-partitions/{logical-partition-id} i.e. <ul style="list-style-type: none"><li>• {logical-partition-id} : aaa97f93-3072-310c-a554-e5c26b1655dd</li><li>• target-name: IBM390PS.T115.BCPE</li></ul>	HWI. <b>TARGET</b> .netid.nau.imagename
Capacity Record i.e. name: MARIANNE	netid.nau.caprec i.e. IBM390PSE.T115.MARIANNE	Object ID /api/cpcs/{cpc-id}/capacity-records/MARIANNE i.e. <ul style="list-style-type: none"><li>• {capacity-record-id} : MARIANNE</li><li>• target-name: IBM390PS.T115. MARIANNE</li></ul>	HWI. <b>CAPREC</b> .netid.nau.caprec
JOB Uri	netid.nau netid.nau.imagename	<i>The 'target-name' used on the asynchronous REST API that returned the JOB Uri (Cap Rec URIs are all synchronous)</i>	HWI. <b>TARGET</b> .netid.nau HWI. <b>TARGET</b> .netid.nau.imagename

# Usage & Invocation – Invocation Example



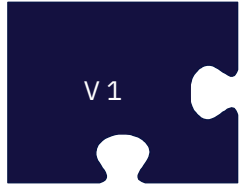
```
/*  
  A previous HWIREST request performed a LIST on CPCs using GET /api/cpcs and stored the uri and target name associated with CPC R32:  
    CPCR32 = "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340"  
    CPCR32TargetName = "IBM390PS.R32"
```

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  
  End  
End
```

# Usage & Invocation – HWILIST vs HWIREST



## HWILIST

- HWI\_LIST\_CPCS

Returned:

*Blank-delimited list of CPC names*

M44 T115



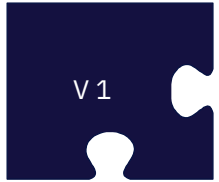
## HWIREST

- 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



## HWICONN

- HWI\_CPC
- 'IBM390PS.M44'

Returned: CPCConnectToken



## 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**)

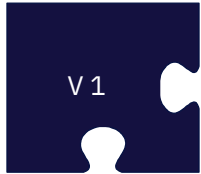
i.e.

- 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



## HWIQUERY

- List of 1 to 64 attributes explicitly specified

- Specified:

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

address bcpii "hwiquery RetCode myConnectToken
myQueryParam. myDiag."
```

Returned:

```
If rc = 0 & Retcode = 0 Then
  Do n=1 to myQueryParam.0
    Say "Parameter"||n||":"myQueryParam.n.ATTRIBUTEVALUE
  End
```



## 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 bcpil "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 bcpil "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



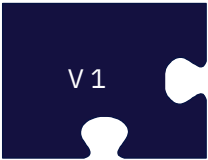
# Usage & Invocation – HWICMD vs HWIREST

---

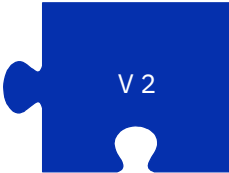
- Many HWICMD/HWICMD2 requests have equivalent HWIREST **POST** */operations* URI requests; a few use a **GET** URI
  - for example:
    - HWI\_CMD\_LOAD -> **POST** /api/logical-partitions/{logical-partition-id}/operations/load
    - HWI\_CMD\_ACTIVATE -> **POST** /api/logical-partitions/{logical-partition-id}/operations/activate
    - HWI\_CMD\_HWMMSG -> **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



- 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)



- 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

Part 2. General services

Chapter 7. General API services

General API services operations summary

Session management services

Request aggregation services

Asynchronous job processing

Query Job Status

Delete Completed Job Status

Cancel Job

Chapter 8. Inventory and metrics services

Inventory services operations summary

Metrics service operations summary

Inventory service

Metrics service

Chapter 9. Metric groups

Monitors dashboard metric groups

Network management metrics

Part 3. CPC management

Chapter 10. Dynamic Partition Manager (DPM)

FICON storage configuration

Operations summary

Partition object

Adapter object

poll at all, the Web Services API also provides asynchronous notification cancellation via its JMS notification capability. IBM recommends that notification facility to determine when a job has ended rather than polling notification" on page 60 for details on this notification mechanism.

If it is not practical for a client application to use asynchronous notification application should introduce elapsed-time delays between successive poll the current job status in order to reduce unproductive use of resources.

**Query Job Status**

The Query Job Status operation returns the status associated with the job.

**HTTP method and URI**

GET /api/jobs/{job-id}

In this request, the URI variable {job-id} is the identifier of an asynchronous job, as returned in the response of the operation that initiated the job.

**Response body contents**

On successful completion, the response body is a JSON object with the following structure:

Field name	Type	Description
status	String Enum	An indication of the current status of the job. The values are as follows: <ul style="list-style-type: none"><li>• "running" - indicates that the job is currently running.</li><li>• "cancel-pending" - indicates that the job has been cancelled but not yet completed.</li><li>• "canceled" - indicates that the job has been cancelled and execution was interrupted.</li></ul>

# 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

---

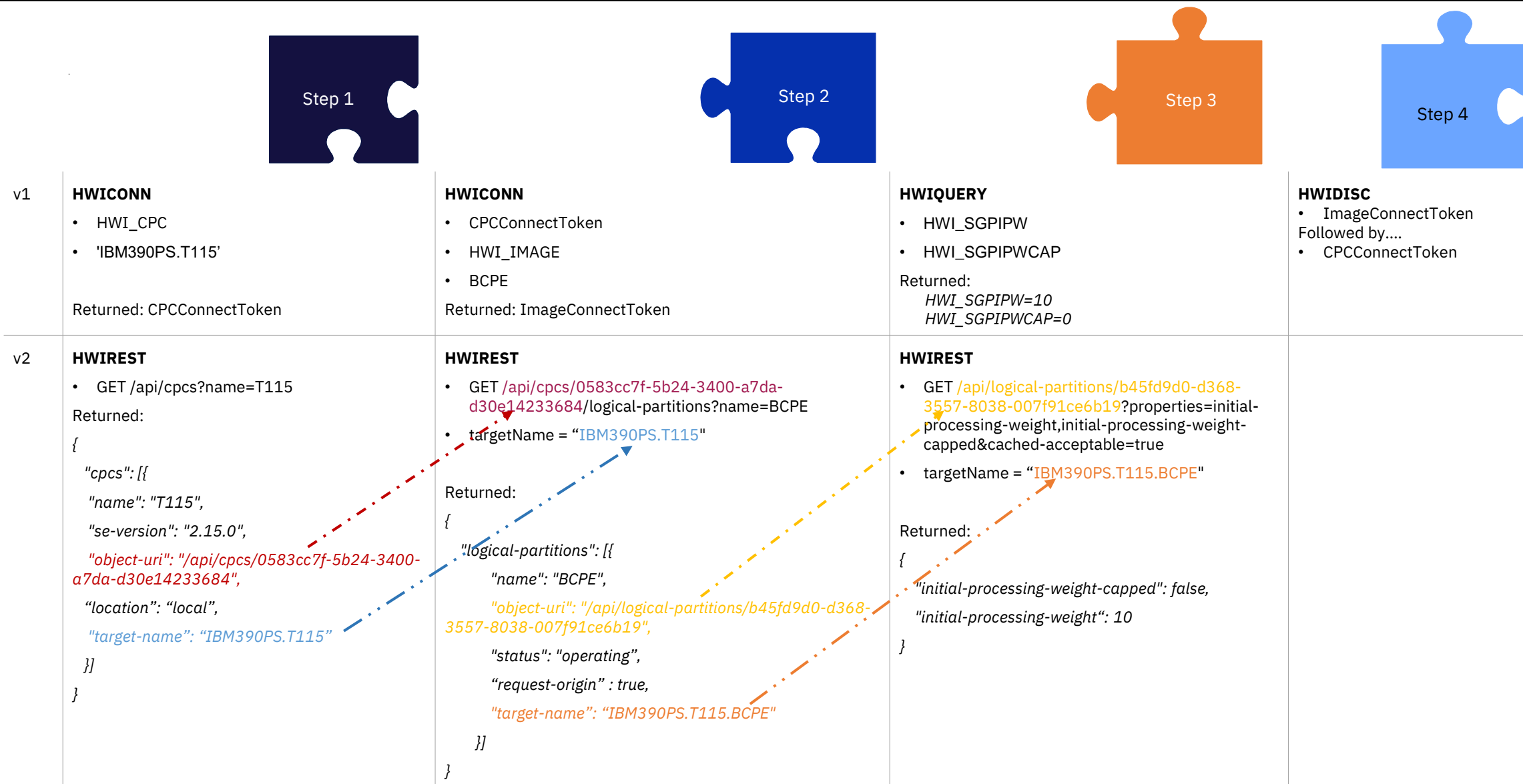
Retrieval of image/LPAR information

- 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



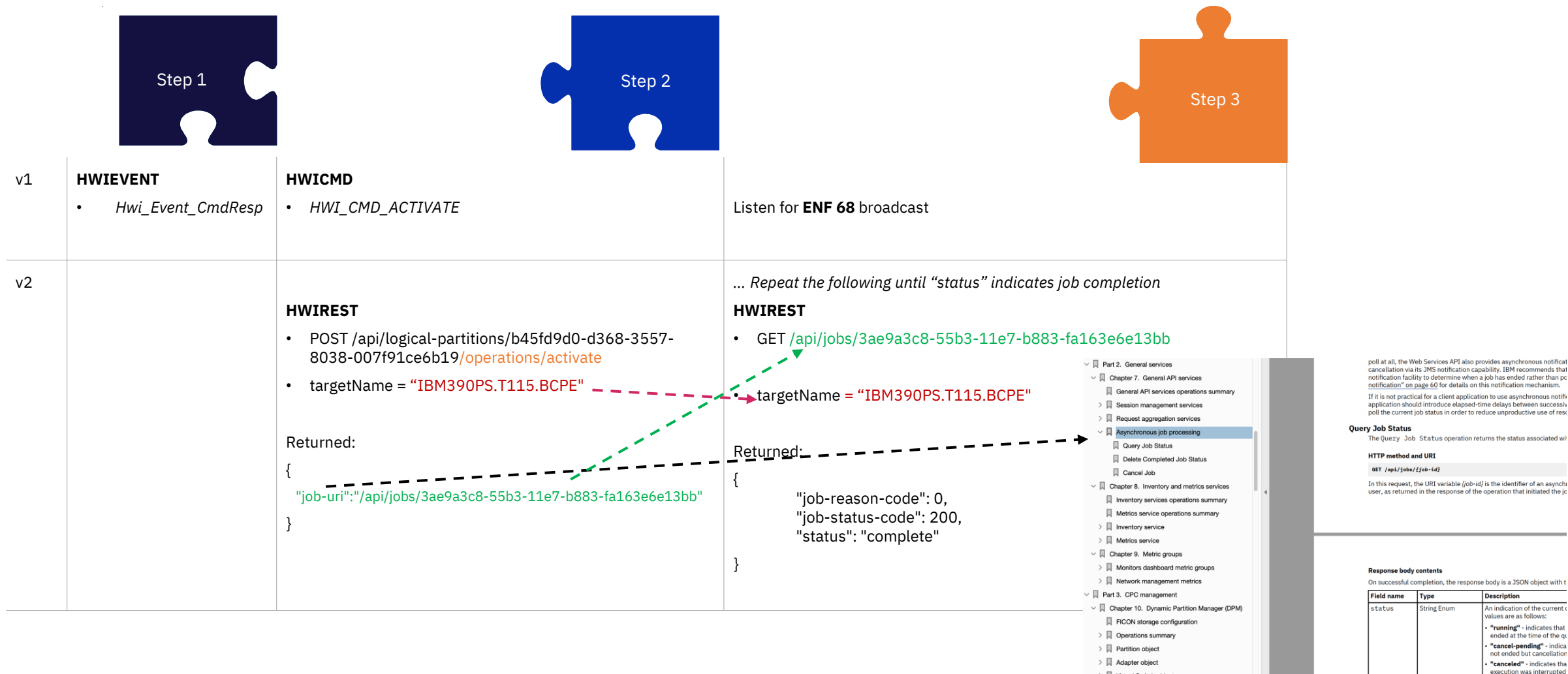
# Usage & Invocation – Comparison – re-activate an LPAR

---

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





# 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>