IBM z/OS BCPii V2R3 Update:
Even greater control of your IBM Z Hardware

Steve Warren
z/OS BCPii Technical Lead
Email: swarren@us.ibm.com
  : @StevieWarr2
August 10, 2017
Agenda

- Quick overview of BCPii
- New BCPii enhancements
  - New BCPii Security Controls (z14)
  - BCPii performance improvements
  - Absolute capping base support
  - Absolute capping group support
  - New more flexible HWICMD2 service
  - Support for larger data returned from SE
  - Support for dynamic CPC name change
  - New HWISET2 service (multiple attribute set)
- BCPii enhancement in V2R2 (SMF logging & SE logging)
Quick overview of z/OS BCPii
Overview - What is BCPii?

**Authorized z/OS application**

- Monitor status or capacity changes
- Obtain configuration data related to CPC or image
- Re-ipl an image
- Change temp. capacity
- Query and update LPAR settings
- Set activation profiles
Quick overview - What is BCPii?

• **Base Control Program** internal interface
  • Allows authorized z/OS applications to have HMC-like control over systems in the process control (HMC) network
  • A set of authorized APIs provided

• **Does not use any external network.**
  • Communicates directly with the SE rather than going over an IP network.

• **A z/OS address space** that manages authorized interaction with the interconnected hardware
Quick overview - Who uses BCPii?

• **z/OS operating system components**
  • System Status Detection (SSD) provided in Sysplex Failure Manager (SFM)
  • Capacity Provisioning Manager (CPM)
  • Hardware Configuration Definition (HCD)

• **Vendor applications**
  • Control center, system management applications
  • Many vendor products use BCPii

• **In-house (customer-written) applications**
Quick overview – BCPii installation steps

- System automatically tries to start BCPii address space at IPL time.
  - You don't need to add anything to COMMNDxx or automation.

- Successful start requires that certain steps have been carried out:
  - Setup on the HMC/SE:
    - Enable Cross Partition Authority for every LPAR that you want to be able to issue or be the target of BCPii commands. (Prior to z14)
    - Enable SNMP and define the Community Name.
      - Both of these can be changed non-disruptively if you wish
  - Setup in z/OS
  - Setup with SAF Security authorizations (in z/OS)
New BCPii Enhancements
Enable "Cross Partition Authority"

Select Save and Change

This should update activation profiles and implement change on active LPAR

Remember that this must be done for every LPAR that will exploit BCPii
Problems with pre-z14 BCPii security mechanism

- “Cross Partition Authority” not granular
  - LPARs that have checkbox checked have unrestricted freedom to perform BCPii functionality against any LPAR that also has its checkbox checked
  - 100% reliant on each LPAR “playing nice” and having BCPii lockdown security in SAF to restrict access to other partitions
    - Unsuitable for service providers installations
    - Unsuitable for installations that have test LPARs that want to enable BCPii to test their applications but don’t have lockdown security

- SHARE Top 50 requirement
  - Close BCPii security exposure

“z/OS 1.11 introduced a powerful new capability whereby XCF can use BCPii to query the status of an LPAR that has stopped updating its heartbeat in the CDS. There are many advantages to this new capability - it enables faster reaction to failed systems increases the likelihood of spin loop recovery being successful and protects you from operator error. To exploit this capability you have to enable BCPii and the only way to control the scope of BCPii actions is currently RACF. However if the service delivery department doesn’t have full control over RACF for every system running in the environment it would be possible for an installation-written program running in one LPAR to impact any LPAR that has been enabled for BCPii.”
New z14 BCPii Security Enhancements

- **Granular system/partition access**
  - Enable/disable send capability per image
  - Enable/disable receive capability per image
    - Enable to receive from all partitions
    - Enable to receive from a list of selected partitions
  - Can limit which images can access CPC via System Details task (HMC or SE)
  - Can limit which images can access other images via Image activation profile and Change LPAR Security task (HMC or SE)

- **System Upgrade migration path**
  - Cross Partition Authority setting remains the same
    - Enabled maps to “BCPii Permission” of send enabled, receive enabled from all partitions
    - Disabled maps to “BCPii Permission” of send/receive disabled.
Defaults to allow the CPC to receive commands from any BCPii-enabled partition.
Z14 BCPii CPC Security Controls – adding partitions

1. Can select from the set of known partitions. Known partitions are from the point of view of the system/partition being configured. “Local” partitions based on the currently defined image profiles. “Remote” partitions learned from managing HMCs.

2. Can enter a partition manually for cases like pre-planning.
z14 BCPii CPC Security Controls – adding partitions

Can also allow only selected partitions.

Check the check box only to remove a CPC/Image from the list.
z14 BCPii LPAR Security Controls – Change LPAR Security Task

- Used to view/alter LPAR BCPii security controls.
- Available on HMC and SE. HMC only supports z14 and higher.
- Click “BCPii Permissions” link to view/alter BCPii security control setting.
z14 BCPii LPAR Security Controls – Image Profile

Security Controls

- Allows BCPii to be enabled on this image and to be able to potentially access other entities.
- Allows BCPii requests to target this partition from all or a select set of local and remote partitions.
z14 BCPii LPAR Security Controls – Multiple Image Profile Security Controls

- Can change multiple image activation profiles at the same time to easily configure the same BCPii permission settings.
z14 BCPii LPAR Security Controls – z/OS Support

- z/OS Migration from z/OS V2R2 and z/OS V2R1 to z/OS V2R3 publication updated
  - See “Enable BCPii communications on the support element” section for more details on the procedure to upgrade your BCPii security on the new z14.
- z/OS BCPii APAR OA53580
  - Update publication z/OS MVS Programming: Callable Services for High-Level Languages to explain the new security controls
  - To fix possible vague error return codes when some obscure security operations are performed.
  - This APAR will be rolled down to V2R1 and higher. (Expect early 4Q17)
BCPii Performance Enhancements
BCPii Performance

- BCPii performance has been an issue for some customers and applications for years
  - Particularly for HWIQUERY and HWILIST operations
    - Especially if querying activation profiles
  - Accentuated if many requests are performed by the application
BCPii Performance Improvements

- BCPii performance improvements when targeting a post-z13 GA2 machine
  - BCPii Performance & Request Enhancements
    - MCL P00339.090 (Bundle S09)
- Significant performance improvements across the board
- z/OS BCPii also leverages hardware update to improve activation profile HWIQUERY and HWILIST performance
  - In base V2R3
  - APAR OA51493 available on V2R1 and V2R2
BCPii Absolute Capping Support
Absolute capping “base” support

• Allows to limit an LPAR’s maximum usage of any CPU type to a specific amount of hardware processor capacity
  • Requires target system to be at zEC12 GA2 and higher
  • Limits only “not dedicated” processors
• New z/OS BCPii support targeting both live image and image activation profile connections for the various engine types available on CPC
  • An LPAR’s Absolute Capping enablement/disablement to be both queried and set for every processor type (GPP, IFL, IIP, etc..)
  • An LPAR’s Absolute Capping value to the hundredth's of a processor
• Available on z/OS V2R1 and higher via APAR OA49720
Absolute capping “group” support

• Allows to limit a group of LPARs of any CPU type to a specific amount of hardware processor capacity (a particular LPAR group)
  • Requires target system to be at z13 GA2 and higher
  • Limits only the “not dedicated” processors
• New z/OS BCPii support targeting live image connections for the various engine types available on CPC
  • An LPAR group’s Absolute Capping enablement/disablement to be both queried and set for every processor type (GPP, IFL, IIP, etc..)
  • An LPAR group’s Absolute Capping value to the hundredth's of a processor
• Available on z/OS V2R3 and higher
New more flexible HWICMD2 service
Current HWICMD service

• Current HWICMD allows the authorized z/OS application to perform major actions against a CPC or image
  • Load, Operating System commands, Activate, Deactivate, etc..
• Parameters are structured as the following:

  CALL HWICMD (ReturnCode, ConnectToken, CmdType, CmdParm_Ptr, DiagArea)
New HWICMD2 service

- `CmdParm_Ptr` points to a fixed-size data structure mapping which is unique for the command desired
  - This allows each possible BCPii command to have a customized “parameter list”, where each command-specific parm is in this unique data structure
  - e.g. If `CmdType = HWI_CMD_LOAD`, the mapping pointed by `CmdParm_Ptr` would look like this:

```c
typedef char    HWI_LOADADDR_TYPE[4];
typedef struct {
    HWI_LOADADDR_TYPE  LoadAddr;
    HWI_LOADP ARM_TYPE LoadParm;
    HWI_FORCE_TYPE     ForceType;
} HWI_CMD_LOADORDUMP_PARM_TYPE;
```
New HWICMD2 service

• What happens when BCPii needs to support a 5-digit load address?
  • It can’t using this parameter list

```c
typedef struct {
    HWI_LOADADDR_TYPE LoadAddr;
    HWI_LOADPARM_TYPE LoadParm;
    HWI_FORCE_TYPE    ForceType;
} HWI_CMD_LOADORDUMP_PARM_TYPE;
```

<- 5th character first char in field
New HWICMD2 service

• HWICMD2 solves this by having the concept of versioned parameter lists
  • New Version number parameter passed into service
  • Version 2 of the parmlist for Load, SCSI Load and SCSI Dump commands will have an extra byte to allow for 5-char IPL device addresses.
  • Eliminates the need for any foreseeable time into the future where the command service will need to be changed again.
  • Available in V2R1 and V2R2 via APAR OA51250
New HWICMD2 service

• The syntax of the new service in non-REXX will be as follows:

CALL HWICMD2(
    ReturnCode,
    ConnectToken,
    CmdType,
    CmdParm_Ptr,
    CmdParmVersion,
    DiagArea);

• CmdParmVersion refers specifies the version of the parameter list
New HWICMD2 service

• Comparison of mapping of version 1 and version 2 parameter list for Load:

/* Version 1 of the Load Command structure */
typedef struct ??<
    HWI_LOADADDR_TYPE LoadAddr;  
    HWI_LOADPARM_TYPE LoadParm;
    HWI_FORCE_TYPE    ForceType;
??> HWI_CMD_LOADORDUMP_PARM_TYPE;

/* Version 2 of the Load Command structure */
typedef struct ??<
    HWI_LOADADDR5_TYPE LoadAddr;  
    HWI_LOADPARM_TYPE LoadParm;
    HWI_FORCE_TYPE    ForceType;
??> HWI_CMD2_LOADORDUMP_PARM_TYPE;
New HWICMD2 service - REXX

• The BCPii REXX interface for the command service will remain unchanged.
• BCPii will invoke HWICMD2 at the highest cmd version level, regardless if REXX program chooses Hwicmd or Hwicmd2
• The syntax of the new service in REXXX remains the same:

```rexx
CmdType = HWI_CMD_OSCmd
CmdParm.PriorityType = Hwi_Cmd_Priority
CmdParm.OSCmdString = 'D GRS'

address bcpii "hwicmd ", ← or hwicmd2
 "ReturnCode ",
 "ConnectToken ",
 "CmdType ",
 "CmdParm. ",
 "DiagArea."
```
Support for larger data returned from SE
BCPii Constraint Relief

• Currently, some larger sized attributes can sometimes exhaust BCPii’s max buffer allowed.
  • BCPii limits the maximum number of data chunks that can come back from the SE.
  •Verbose attributes (especially ones using XML) such as HWI_EC_MCL_INFO most susceptible to reaching the arbitrary BCPii limit.
• z/OS BCPii will now handle virtually any sized data returning from SE.
• No application changes required.
• Available on V2R1 and V2R2 via APAR OA53268 around the end of October, 2017.
Support for Dynamic CPC name change
Dynamic CPC Name Change

• Today, z/OS BCPii does not react properly when an installation dynamically changes the name of their CPC and reboots the SE to make the change effective.

• BCPii has no easy way of knowing that a request targeting one CPC physical machine could suddenly now be arriving at another CPC physical machine.
Dynamic CPC Name Change

- HWICONN CPCX
- HWIQUERY CPCX
- HWIQUERY CPCX

Process Control (HMC) Network
Dynamic CPC Name Change

- BCPii provides support for changing the name of a CPC with ACTIVE images.
- When a CPC name change is detected, BCPii takes the following actions:
  - Invalidates outstanding connections to the affected CPC.
  - Issues an ENF68 informing interested parties of the name change ((hardware event HWIENF68_HWEVENT_NAMECHG)
  - Reconnects to the local CPC (if the local CPC name is changed)
- Applications targeting the CPC using the old name will get a return code indicating the connection is no longer valid (e.g, HWI_CONNECT_TOKEN_INV or HWI_TARGET_CPC_CHANGED).
Dynamic CPC Name Change

- **HWI_TARGET_CPC_CHANGED** return code
  - **Meaning:** The CPC name represented by the specified token is valid, but does not represent the same physical machine targeted by the initial HWICONN call. All connections established prior to the name change can no longer be used.
  - **Action:** The application should cease from using this connect token. If the application intends to target the CPC using the name represented by the specified connect token, they must first re-connect to the CPC before issuing any BCPii service call.
New HWISET2 (multiple attribute set) service
BCPii Set service (HWISET) functionality

• One CPC, image, or activation profile attribute can be set at a time
  • Higher latency connection between z/OS and the SE means that there are short time lags between each HWISET call
  • Configuration likely to have temporarily incompatible values set for attributes

• Example:
  • Image A – defined capacity value (HWI_DEFCAP) = 10
  • Image B – defined capacity value = 5
  • Want to set Image A to 8 and Image B to 8.
  • Application issues
    • HWISET image A – HWI_DEFCAP = 8
    • HWISET image B – HWI_DEFCAP = 8
  • There is a short time period of configuration inconsistency, where A is 8 and B is 5, when the end goal is 8 and 8.
New HWISET2 (Multiple attribute set) service

- Multiple attributes can be set at the same time
  - Provides better adjustments to the hardware configuration
  - Shorter periods of configuration
- Example:
  - Image A – defined capacity value (HWI_DEFCAP) = 10
  - Image B – defined capacity value = 5
  - Want to set Image A to 8 and Image B to 8.
- Application issues
  - HWISET2 with HWI_DEFCAP = 8 for image A and HWI_DEFCAP = 8 for image B
  - Much shorter period of configuration inconsistency. When call returns to application, both values have been adjusted.
HWISET2 particulars

• New HWISET2 service similar to the HWIQUERY interface for querying multiple attributes on a single call
• Additional benefit of having a “two-phase commit” like behavior where either all changes will be made or none of the changes will be made when targeting at z13 GA2 or higher machine.
• HWISET2 can target z13 GA1 and lower successfully.
  • If a failure occurs with one of the attributes, a return code of HWI_SET2_PARTIAL_UPDATE will be returned along with an indicator of the first failing attribute and the reason for the failure.
  • User will either have to manually back out the attributes already updated or attempt to rectify the partial update in another manner.
HWISET2 particulars (continued)

• All resources specified must be located on the same CPC.
  • Each attribute to be set specifies a ConnectToken representing that resource. The ConnectToken must either be:
    • Identical to the CPC ConnectToken on the HWISET2 parameter list
      • i.e. the CPC attribute is being set
    • A child connection of the CPC ConnectToken on the HWISET2 parameter list.
      • i.e. the ConnectToken is an image or activation profile on that CPC
HWISET2 – Syntax

The syntax of the new service in non-REXX will be as follows:

```
CALL HWISET2(
    ReturnCode,
    InConnectToken,
    SetParm_Ptr,
    NumOfAttributes,
    DiagArea);
```

InConnectToken must represent a BCPii CPC connection.
SetParm_Ptr points to an array of Set structures.
NumofAttributes specifies the # of elements in the array (up to 9 attributes can be set at one time).
HWISET2 Data structure

- **SetParm_Ptr** points to a structure mapped by SetParm

```c
typedef struct {
    HWI_CONNTOKEN_TYPE Set2_CToken;
    int Set2_SetType;
    char * Set2_SetValue_Ptr;
    int Set2_SetValueLen;
}HWI_SET2_SETPARAM_TYPE;
```
HWISET2 – Syntax (REXX)

• The BCPIi REXX interface for the SET2 service will use constant stem variable tails ala HWIQUERY to set the various set values required.

```
stemName.0 = 3
stemName.1.SET2_Ctoken = <image1ConnToken on CPC1>
stemName.1.SET2_SetType = HWI_DEFCAP
stemName.1.SET2_SetValue = 5
stemName.2.SET2_Ctoken = <image2ConnToken on CPC1>
stemName.2.SET2_SetType = HWI_DEFCAP
stemName.2.SET2_SetValue = 3
stemName.3.SET2_Ctoken = <imageActProfConnToken on CPC1>
stemName.3.SET2_SetType = HWI_DEFCAP
stemName.3.SET2_SetValue = 5
```

• Note that the connect tokens must all be targeted to the same CPC as specified on the InConnectToken parameter.
HWISET2 new return codes and abend codes

- **New return codes:**
  - HWI_SET2_SETPARM_INACCESSIBLE – setparm structure not addressable
  - HWI_SET2_NUMOFATTRIB_INV – specified a value $< 1$ or $> 9$
  - HWI_SET2_CONNECT_TOKEN_INV – connect token specified in the data structure either not associated with InConnectToken or bad value
  - HWI_SET2_PARTIAL_UPDATE – mentioned on previous slide

- **New abend codes:**
  - HWI_SET2_PRIM_ADDR – parms not in primary AS
  - HWI_SET2_NOT_ADDR – parms not addressable
  - HWI_SET2_BAD_PARM – number of parms bad
BCPii Enhancement in V2R2 – BCPii SMF Logging
The need for SMF support in z/OS BCPii

- **Problem Statement / Need Addressed**
  - Whenever a BCPii application issues an API that modifies hardware resources, there are not sufficient audit trails to keep track of which application/user modified the resources (SHARE Top 50 requirement SSMVSE12018)

- **Solution**
  - BCPii now cuts SMF 106 records for **successful** HWISET and HWICMD API calls

- **Benefit / Value**
  - Sufficient audit information to know what resources were modified by BCPii applications
BCPii SMF support – How to cut the records

• New SMF 106 record
  • IEASYSxx points to SMFPRMxx member
  • SMFPRMxx member
    • SYS(TYPE(106))

• 2 subtypes
  • Subtype 1 (HWISET) SYS(TYPE(106(1)))
    • Cuts detailed information about each successful HWISET call
  • Subtype 2 (HWICMD) SYS(TYPE(106(2)))
    • Cuts detailed information about each successful HWICMD call
BCPii SMF support – How to read the records

• Mapped by BCPii SMF Type 106 mapping macro
  • SYS1.MACLIB(HWISMF6A)

• Supplied sample formatting JCL
  • HWI6AFMT
    • Copies BCPii SMF Type 106 records from a data set or logstream to a temporary dataset
    • Using the DFSORT-provided ICETOOL:
      • Sorts the type 106, subtype 1 and subtype 2 records
      • Produces a summary report for the type 106 records
      • Produces a detailed report for type 106 for subtype 1 and subtype 2

• HWIRPTMP
  • SMF type 106 JCL variable map as input to the DFSORT-provided ICETOOL
What is actually in the SMF Data

- What is actually cut by BCPii in the SMF Record Type 106?
  - Connection Type of the HWISET or HWICMD request
    - CPC, Image, Reset activation profile, Image activation profile, or Load activation profile
  - CPC Name
  - Request parameter
    - Either the image name or activation profile name specified by the requester
  - ASID
  - Job name
  - User ID
What specifically is in SMF records for HWISET?

• Detailed information in Subtype 1 (HWISET):
  • SetType
    • Resource that was modified
  • Set Type Value Length
    • Length of the value being set
  • Set Parameter
    • The actual value being set
What specifically is in SMF records for HWICMD?

- Detailed information in Subtype 2 (HWICMD):
  - Command Type
  - Command that was issued
  - Command parameter list passed to BCPii
  - Optional XML data sent on request
### Sample report output:

**SUBTYPE 1 RECORDS SUMMARY REPORT**  
02/03/15 10:18:52

<table>
<thead>
<tr>
<th>CPC Name</th>
<th>IBM390PS.H87</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEN</td>
<td>397</td>
</tr>
<tr>
<td>SEG</td>
<td>0</td>
</tr>
<tr>
<td>FLG</td>
<td>DE</td>
</tr>
<tr>
<td>RTY</td>
<td>106</td>
</tr>
<tr>
<td>TME</td>
<td>10:45:05</td>
</tr>
<tr>
<td>DTE</td>
<td>2015/02/03</td>
</tr>
<tr>
<td>SID</td>
<td>BCPJ</td>
</tr>
<tr>
<td>WID</td>
<td>JES2</td>
</tr>
<tr>
<td>STP</td>
<td>1</td>
</tr>
</tbody>
</table>

**VERSION PROD NAME MVS PROD SYSTEM NAME CONN TYPE CPC**  
01 BCPII SP7.2.2 BCPJ 2 IBM390PS.H87

**RTN ASD JOBNAME USER SET TYPE SET PARM LEN**  
LP8 2D HWASEY04 SWARREN 116 4

**TDA**  
000003E700000000000000000000000000000000
Usage & Invocation

- Sample report output:

**SUBTYPE 2 RECORDS SUMMARY REPORT** 02/13/15 14:47:02 - 1

**CPC Name** IBM390PS.H87

<table>
<thead>
<tr>
<th>LEN</th>
<th>SEG</th>
<th>FLG</th>
<th>RTY</th>
<th>TME</th>
<th>DTE</th>
<th>SID</th>
<th>WID</th>
<th>STP</th>
</tr>
</thead>
<tbody>
<tr>
<td>3955</td>
<td>0</td>
<td>DE</td>
<td>106</td>
<td>14:34:05</td>
<td>2015/02/13</td>
<td>BCPJ</td>
<td>JES2</td>
<td>2</td>
</tr>
</tbody>
</table>

**VERSION** **PROD NAME** **MVS PROD** **SYSTEM NAME** **CONN TYPE** **CPC**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>BCPII</td>
<td>SP7.2.2</td>
<td>BCPJ</td>
<td>1</td>
<td>IBM390PS.H87</td>
</tr>
</tbody>
</table>

**RTN** **ASID** **JOBNAME** **USER** **COMMAND TYPE**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-----------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>--------</td>
</tr>
<tr>
<td>47</td>
<td>HWAECX16</td>
<td>IBMUSER</td>
<td>14</td>
<td></td>
</tr>
</tbody>
</table>

**CMD XML or IPLToken**

0000000123DE04100000005C00000000
Sample output

- Sample report output:

```
HWI_CMD_TEMPCAP SMF RECORDS REPORT      02/13/15
14:47:02   - 1

<table>
<thead>
<tr>
<th>LEN</th>
<th>SEG</th>
<th>FLG</th>
<th>RTY</th>
<th>TME</th>
<th>DTE</th>
<th>SID</th>
<th>WID</th>
<th>STP</th>
</tr>
</thead>
<tbody>
<tr>
<td>3955</td>
<td>0</td>
<td>DE</td>
<td>106</td>
<td>14:34:05</td>
<td>2015/02/13</td>
<td>BCPJ</td>
<td>JES2</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CTY</th>
<th>TEMPCAPTYPE</th>
<th>CAPXMLPTR</th>
<th>CAPXMLSIZE</th>
<th>XMLData</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td></td>
<td>23DE0410</td>
<td>5C</td>
<td>&lt;add&gt;&lt;recordid&gt;0</td>
</tr>
</tbody>
</table>
```
Enabling BCPii SMF recording

- To activate BCPii SMF recording:
  - Parmlib method
    - Add the necessary statements to the SMFPRMxx parmlib member for SMF Type 106
      - SYS(TYPE(106))
      - SYS(TYPE(106(1)))
      - SYS(TYPE(106(2)))
      - SYS(TYPE(106(1:2)))
    - Issue the SET SMF=xx command to activate the parmlib changes
  - On the fly method
    - SETSMF command
More Advanced BCPii Programming – Debugging Programming Errors

- BCPii API request history logging available on z13 GA2
  - MCL P00339.185 (Bundle S20)
  - Shows BCPii requests coming into the SE
- API Request Log Option
  - Single Object Operations → Support Element Management → Audit and Log Management task (in the security section)
  - Select Api Request checkbox
## More Advanced BCPii Programming – Debugging Programming Errors

### Api Request

<table>
<thead>
<tr>
<th>Display Identifier</th>
<th>Request Type</th>
<th>Start Time</th>
<th>Finish Time</th>
<th>Requester</th>
<th>Request Status</th>
<th>Protocol</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>41,014</td>
<td>GET</td>
<td>June 6, 2016</td>
<td>June 6, 2016</td>
<td>GDPSSNMP</td>
<td>Success</td>
<td>BCPii</td>
<td>1.3.6.1.4.1.2.6.42.1.0.151.0.1374470907</td>
</tr>
<tr>
<td>41,016</td>
<td>GET</td>
<td>June 6, 2016</td>
<td>June 6, 2016</td>
<td>GDPSSNMP</td>
<td>Success</td>
<td>BCPii</td>
<td>1.3.6.1.4.1.2.6.42.2.0.1.0.225325236</td>
</tr>
<tr>
<td>41,017</td>
<td>GET</td>
<td>June 6, 2016</td>
<td>June 6, 2016</td>
<td>GDPSSNMP</td>
<td>Success</td>
<td>BCPii</td>
<td>1.3.6.1.4.1.2.6.42.1.0.161.0.1374470907</td>
</tr>
</tbody>
</table>
BCPii Reference Materials

- z/OS 2.2 MVS Programming: Callable Services for High-Level Languages
  - Primary BCPii documentation including installation instructions and BCPii API documentation (including BCPii REXX support)
- z/OS 2.2 MVS System Commands:
  - START HWISTART and STOP HWIBCPII commands.
- z/OS 2.2 MVS Diagnosis: Tools and Service Aids:
  - BCPii's CTRACE documentation.
- z/OS 2.2 MVS Programming: Authorized Assembler Services Reference, Volume 2 (EDT-IXG)
  - BCPii's ENF68 documentation.
- z/OS 2.2 MVS System Management Facilities (SMF)
  - BCPii's primary SMF documentation on SMF Type 106 records
- z/OS 2.2 MVS Initialization and Tuning Reference
  - Miscellaneous documentation
- z/OS 2.2 MVS System Codes
  - BCPii abend '042'x documentation
Yet More BCPii Information!

• Other SHARE presentations regarding BCPii:
  • IBM z Systems Hardware Management Console (HMC) 2.13.1, March 7, 2017 presentation
• Cheryl Watson’s Tuning Letter, 2013, No.2
  • Focus: Exploiting z/OS, Part 3 - BCPii
• IBM Redbooks (http://www.redbooks.ibm.com)
  • System z Parallel Sysplex Best Practices
Yet More BCPii Information (Older but useful info)!

• z/OS Hot Topics
  • August 2013: Quick and Easy: BCPii (pg. 63)
  • August 2012: Seeing BCPii with new eyes (pg. 7)
  • August 2009: The application doesn’t fall far from the tree (BCPii: Control your HMC and support element directly from z/OS apps)
Quick and Easy: BCP!1!

BY STEVE WARREN AND RISA MDELL

It's time to check out the Basic Control Program interface (BCP!) in z/OS Version 2 Release 1 (V2R1). The improvements in BCP! function might be the quick and easy recipe to help you start using this basic function of the z/OS operating system. If you are already using BCP!, you can now use it more efficiently than ever.

BCP! at your service
In z/OS V2R1, BCP! supports applications written in the REXX programming language, known for its ease of use. BCP! also maximized the traffic to the support elements (SE). Less traffic to the SE means equal improved performance for you. Let's first take a step back and look at BCP! BCP! is a tool to access System z hardware, external to any z/OS authorized applications running in any address space. For example, you might want to:

- Find out what is going on with the hardware
- Perform powerful tasks like re-JES or load an LPAR
- Reserve monitors when certain hardware errors occur

Do you want to do all these things from the convenience of your z/OS applications? If so, BCP! is at your service. It's not necessary to install a name of products or complete a complicated install process to start using it.

Ready for REXX?
In z/OS V2R1, the BCP! APIs support applications using either the C or assembly programming languages. Over the years, there has been a growing and vocal demand for REXX programming language support in BCP! API.

We listened and delivered
In z/OS V2R1, the BCP! support for REXX and a much simpler programming model than either the C or assembler programming languages, you can get applications up and running quickly and easily.

BCP! APIs support applications written in the z/OS System REXX, TSO/SE REXX, and independent software vendor (ISV) provided REXX programming environments. Not only does writing with the REXX programming language allow you to develop BCP! applications in record time, but also maintains your investment in your existing BCP! applications written in C or assembly. These REXX applications can work right alongside them.

Staple BCP! REXX ease
Here is a sample BCP! REXX code that lists all the interconnected processors in your Hardware Management Console (HMC) network.

Notice in the previous processing style, how quickly the list is and valid. BCP! returns the data in a main variable. The -1 element of the array variable contains the number of items returned and the 0 to 1 elements contain the actual names of the processors connected to the system. This is not an example, but the other BCP! APIs calls are just as intuitive and easy to use.

Figure 1. Sample BCP! REXX ease

\[ \text{COPYRIGHT} \text{2013 by SHARE Inc. Except where otherwise noted, this work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 license.} \text{http://creativecommons.org/licenses/by-nc-nd/3.0/} \]
**BCPi Blog**

- Great new way to get tips, insight and the latest BCPii technical information
  - Hosted on IBM Mainframe Insights

and

- [http://mainframeinsights.com/?s=BCPi](http://mainframeinsights.com/?s=BCPi)
 Some blog entries:

- Tips for adding Temporary Capacity with z/OS BCPii
- Putting our clients first: The z/OS BCPii journey
- z/OS BCPii and REXX: A cool combo for automation
- BCPii and REXX: Walking arm in arm
- The wait is over! Improved performance for BCPii’s HWILIST and HWIQUERY services
- Top 10 questions from BCPii customers
- We heart z/OS BCPii
- How about a slice of BCPii? (Discussion of BCPii samples)
- A slice of pizza, a cup of coffee and a quick SSDPP…
- Steve Warren, z/OS BCPii Technical Lead, Answers Your Questions
The BCPII HWICMD service provides a command called TEMPCAP (or Temporary Capacity), which gives you the ability to add or remove processors in a CPC on a temporary basis using profiles. This functionality assumes that you have purchased capacity records, which allow your program to activate extra dormant CPUs on your machine to increase the capacity of the CPC.

This is especially useful during peak workloads when you may need a little more.
Questions?