How to Make the Most out of BCPii

Steve Warren
IBM
swarren@us.ibm.com

August 6th, 2014
Session 15579

#SHAREorg

SHARE in Pittsburgh 2014

Trademarks

Notice Regarding Specialty Engines (e.g., zIIPs, zAAPs and IFLs):

Any information contained in this document regarding Specialty Engines ("SEs") and SE eligible workloads provides only general descriptions of the types and portions of workloads that are eligible for execution on Specialty Engines (e.g., zIIPs, zAAPs, and IFLs). IBM authorizes customers to use IBM SE only to execute the processing of Eligible Workloads of specific Programs expressly authorized by IBM as specified in the “Authorized Use Table for IBM Machines” provided at www.ibm.com/systems/support/machine_warranties/machine_code/aut.html (“AUT”).

No other workload processing is authorized for execution on an SE.

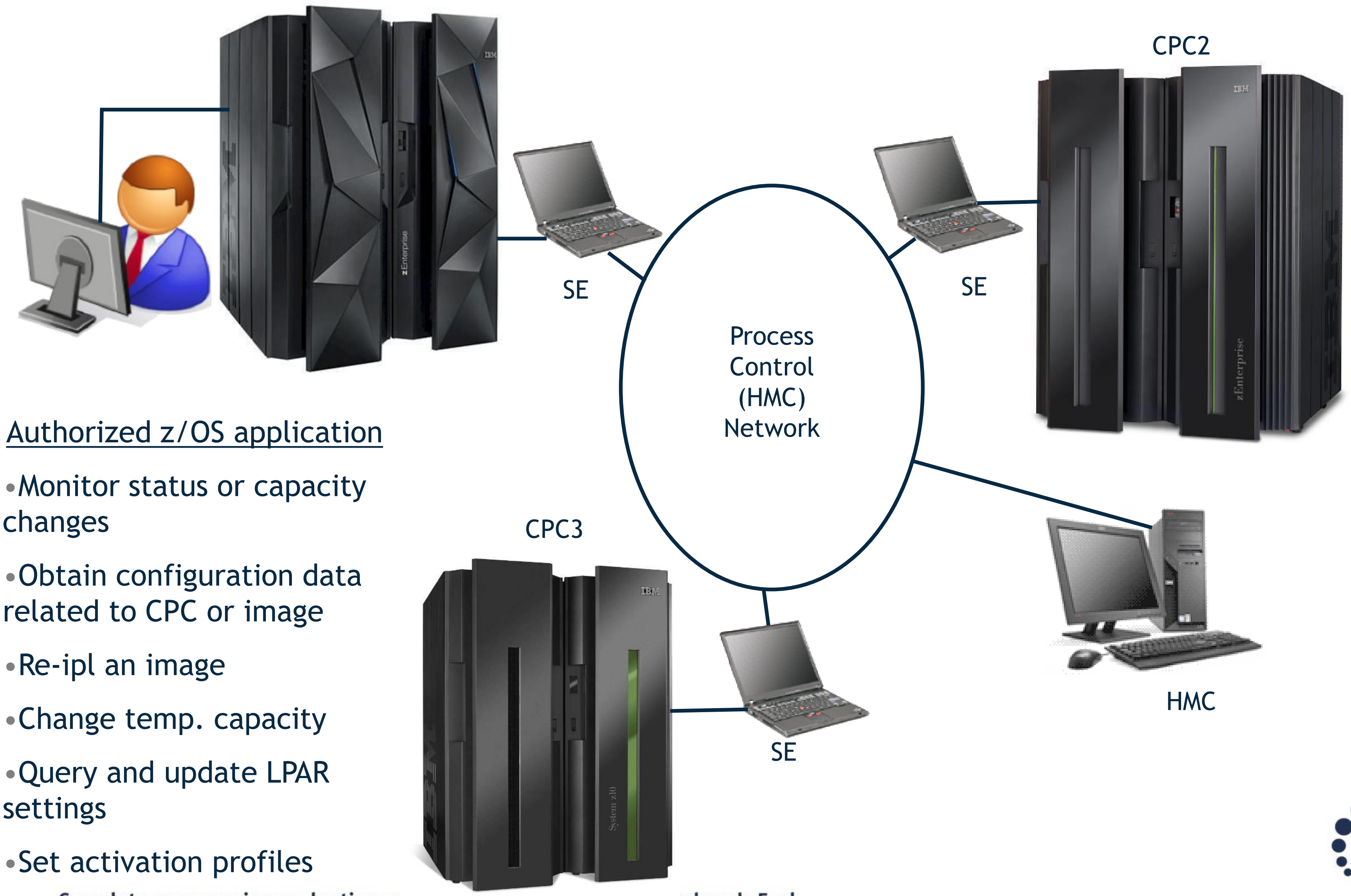
IBM offers SEs at a lower price than General Processors/Central Processors because customers are authorized to use SEs only to process certain types and/or amounts of workloads as specified by IBM in the AUT.

Agenda

- Quick BCPii Overview
  - What is it and what can I do with it?
  - Installation basics
- What can I do with BCPii?
  - Off the shelf solution examples
    - IBM offerings
    - Non-IBM offerings
  - Writing my own BCPii app
    - Typical BCPii usage when writing your own application
    - Programming basics
- Reference material

Quick BCPii Overview

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

Overview - What is BCPii?

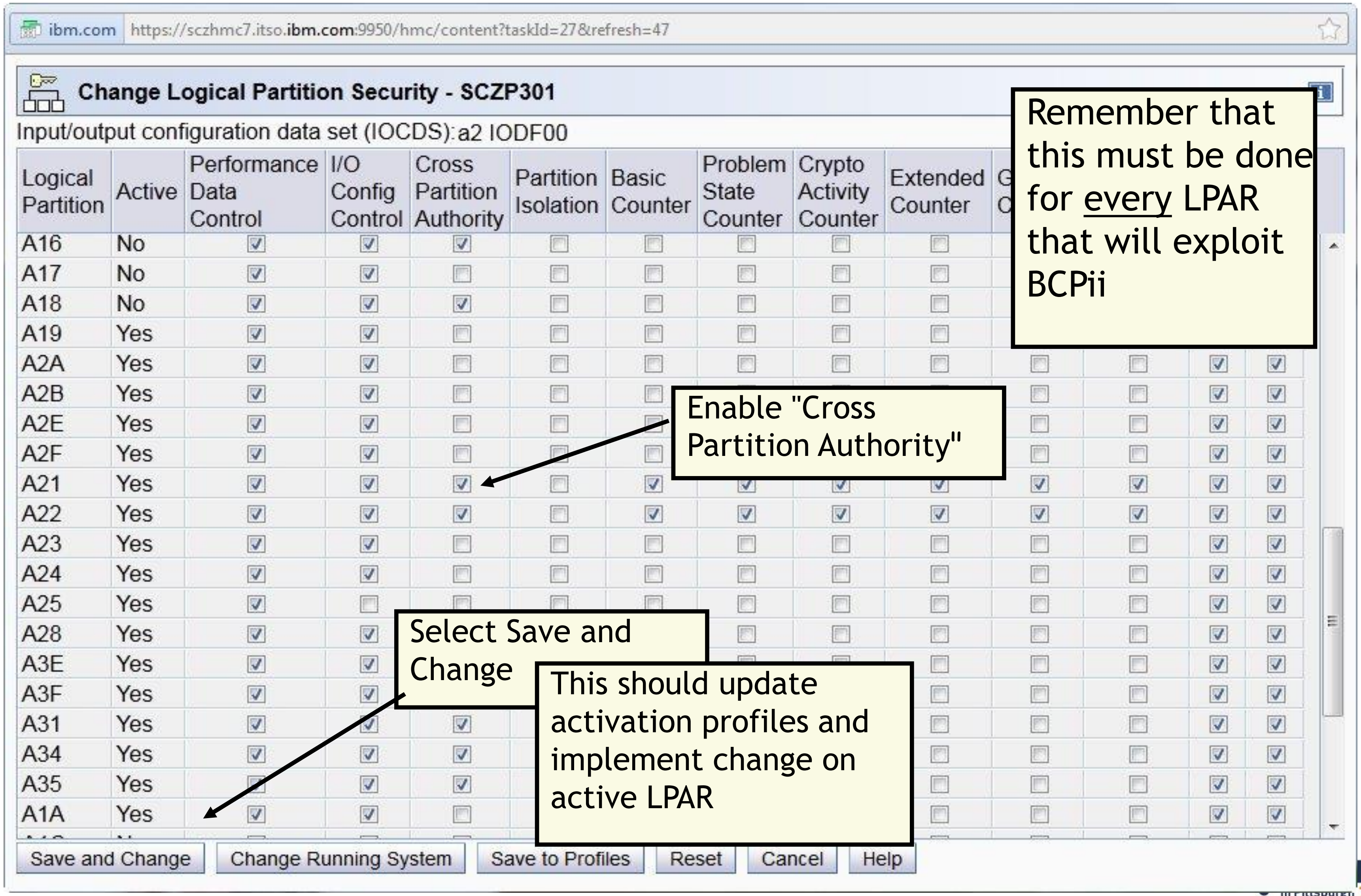
- Base Control Program internal interface
  - Allows authorized z/OS applications to have HMC-like control over systems in the process control (HMC) network
  - A set of authorized APIs provided
- Does not use any external network
  - Communicates directly with the SE rather than going over an IP network
- A z/OS address space that manages authorized interaction with the interconnected hardware

Installation Basics

- Starting with z/OS 1.11, 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.
    - 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)

Setup on the SE/HMC – Part 1


ibm.com https://sczhmc7.itso.ibm.com:9950/hmc/content?taskId=27&refresh=47

Change Logical Partition Security - SCZP301

Input/output configuration data set (IOCDS): a2 IODF00

Logical Partition	Active	Performance Data Control	I/O Config Control	Cross Partition Authority	Partition Isolation	Basic Counter	Problem State Counter	Crypto Activity Counter	Extended Counter	G C			
A16	No	✓	✓	✓	☐	☐	☐	☐	☐				
A17	No	✓	✓	☐	☐	☐	☐	☐	☐				
A18	No	✓	✓	✓	☐	☐	☐	☐	☐				
A19	Yes	✓	✓	☐	☐	☐	☐	☐	☐				
A2A	Yes	✓	✓	☐	☐	☐	☐	☐	☐	☐	☐	✓	✓
A2B	Yes	✓	✓	☐	☐	☐				☐	☐	✓	✓
A2E	Yes	✓	✓	☐	☐	☐				☐	☐	✓	✓
A2F	Yes	✓	✓	☐	☐	☐				☐	☐	✓	✓
A21	Yes	✓	✓	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓
A22	Yes	✓	✓	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓
A23	Yes	✓	✓	☐	☐	☐	☐	☐	☐	☐	☐	✓	✓
A24	Yes	✓	✓	☐	☐	☐	☐	☐	☐	☐	☐	✓	✓
A25	Yes	✓	☐	☐	☐	☐	☐	☐	☐	☐	☐	✓	✓
A28	Yes	✓	✓				☐	☐	☐	☐	☐	✓	✓
A3E	Yes	✓	✓						☐	☐	☐	✓	✓
A3F	Yes	✓	✓						☐	☐	☐	✓	✓
A31	Yes	✓	✓	✓					☐	☐	☐	✓	✓
A34	Yes	✓	✓	✓					☐	☐	☐	✓	✓
A35	Yes	✓	✓	✓					☐	☐	☐	✓	✓
A1A	Yes	✓	✓	☐					☐	☐	☐	✓	✓

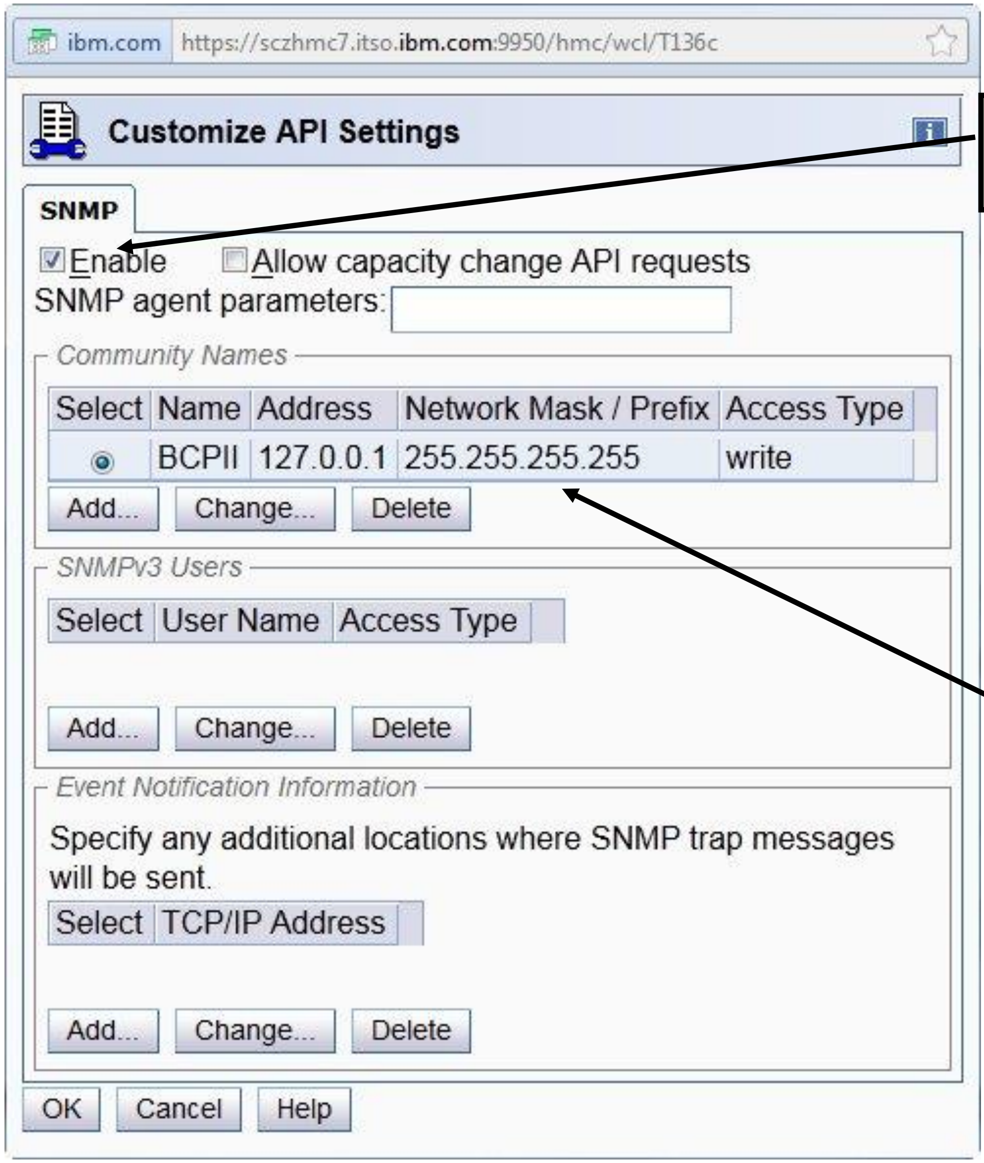
Remember that this must be done for every LPAR that will exploit BCPii

Enable "Cross Partition Authority"

Select Save and Change

This should update activation profiles and implement change on active LPAR

Setup on the SE/HMC – Part 2



ibm.com https://sczhmc7.itso.ibm.com:9950/hmc/wcl/T136c

Customize API Settings

SNMP

[✓] Enable [ ] Allow capacity change API requests

SNMP agent parameters:

Community Names

Select	Name	Address	Network Mask / Prefix	Access Type
●	BCPII	127.0.0.1	255.255.255.255	write

SNMPv3 Users

Select	User Name	Access Type

Event Notification Information

Specify any additional locations where SNMP trap messages will be sent.

Select	TCP/IP Address

Enable "SNMP APIs"

Add a "community name" associated with BCPii's "port" into the SE

Setting up BCPii in z/OS

- BCPii address space requirements
  - hlq.SCEERUN and hlq.SCEERUN2 data sets must be in LNKLST.

Setting up BCPii Security definitions

- General BCPii authority for applications
  - The profile HWI.APPLNAME.HWISERV in the FACILITY resource class controls which applications can use BCPii services.
    - Anyone wishing to use BCPii must at least have READ access to this profile.
  - The FACILITY class must be RACLISTed.

Setting up BCPii Security definitions

A BCPii application needs to have authority to the particular resource (CPC, Image, Capacity Record, Activation Profile) that it is trying to access (This is IN ADDITION to having access to the HWISERV FACILITY profile).

Profile names are:

- CPC: HWI.TARGET.netid.nau
- Image: HWI.TARGET.netid.nau.imagename
- Capacity Record: HWI.CAPREC.netid.nau.caprec
- Activation Profile: HWI.TARGET.netid.nau

Note: netid.nau is the 3-17 character SNA name for CPC (defined when you first define the SE to the HMC)

Setting up BCPii Security Definitions

- Choose SAF authority to a specific resource listed above to allow BCPii service access:
  - HWILIST, HWICONN, HWIDISC, HWIEVENT, HWIQUERY
    - At least READ access
  - HWISET
    - At least UPDATE access
  - HWICMD
    - At least CONTROL access

Setting up BCPii Security Definitions – Associate CPC definintion with community name

- When defining the CPC profiles, APPLDATA must match the community name you specified on the SE:
  - `RDEFINE FACILITY HWI.TARGET.USIBMSC.SCZP301 UACC(NONE) APPLDATA(‘BCPII’)`

The BCPii address space

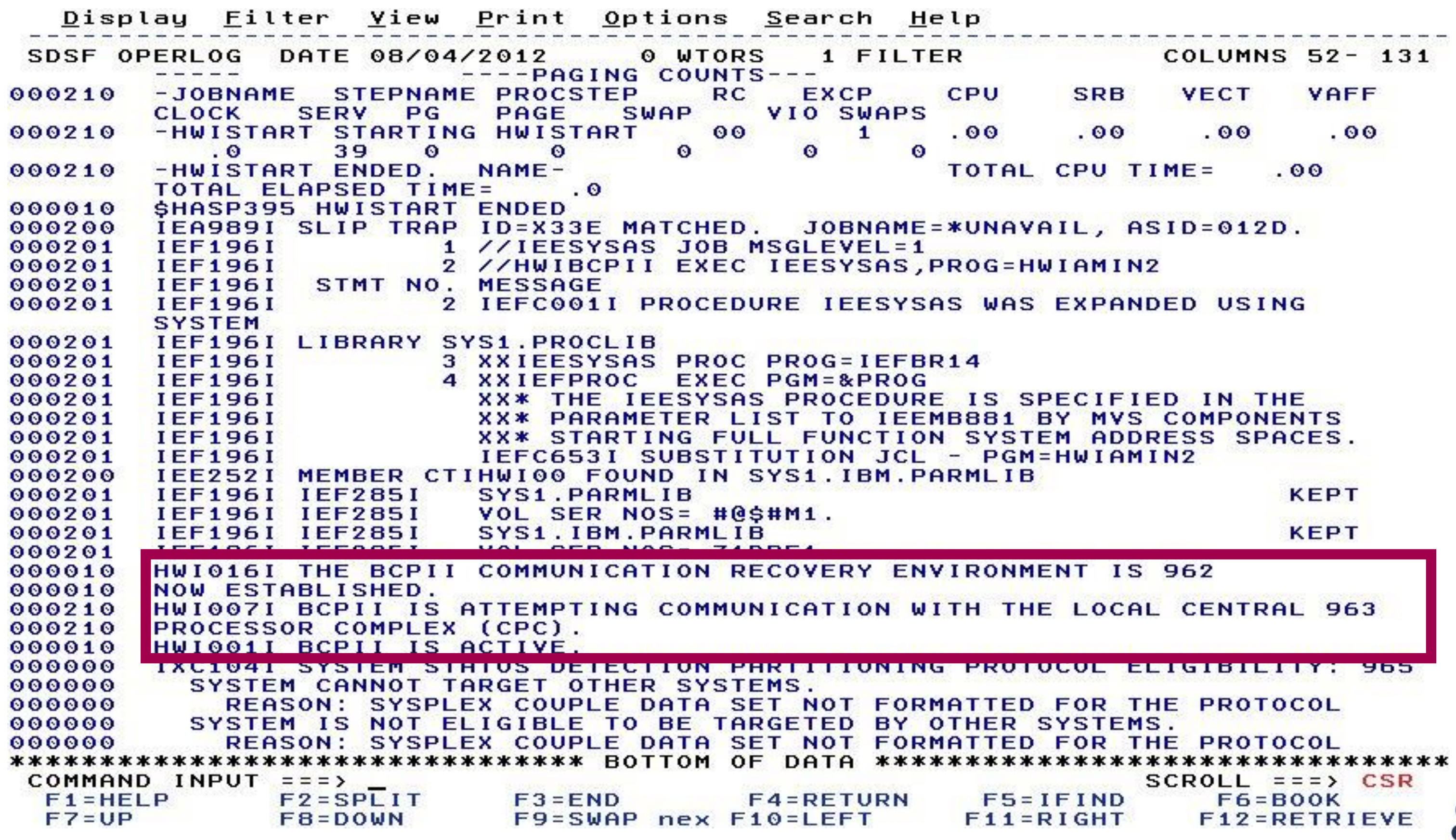
- **System automatically tries to start BCPII address space at every IPL:**
  - Address space name is HWIBCPII.
  - Address space shows up in SDSF DA, but not in D A,L output.
- **Address space can be stopped using P HWIBCPII command:**
  - Once the address space is stopped, no BCPII calls will be processed.
  - ENF signal is broadcast to let any interested parties know that the interface is stopping.
  - If P command doesn't work, you can use a CANCEL HWIBCPII
- **Address space can be started again using S HWISTART (HWISTART is delivered in SYS1.PROCLIB)**

How to check the status of BCPii

- There is currently no console command to check the status of BCPii.
- If Pre-reqs are not in place at IPL time, address space will start and then stop.
- So, if address space is active, that is at least a positive sign.
  - Check for message HWI001I BCPII IS ACTIVE among IPL messages
  - Doesn't guarantee that every CPC has been set up to support BCPII
  - Currently the only way to check is from a program that uses the BCPII API
    - If program doesn’t get ‘F00’X return code (HWI_NOT_AVAILABLE), BCPii is active on the system.

Start BCPii

Having completed the setup work on the local CPC and in RACF, we now start BCPii address space:



```
  Display  Filter  View  Print  Options  Search  Help
 -------------------------------------------------------------------------------
 SDSF OPERLOG  DATE 08/04/2012      0 WTORS    1 FILTER         COLUMNS 52- 131
       -----            ----PAGING COUNTS---
000210  -JOBNAME  STEPNAME PROCSTEP    RC    EXCP    CPU     SRB    VECT    VAFF
        CLOCK    SERV   PG    PAGE    SWAP    VIO SWAPS
000210  -HWISTART STARTING HWISTART    00       1    .00     .00     .00     .00
           .0    39    0      0      0      0     0
000210  -HWISTART ENDED.   NAME-                     TOTAL CPU TIME=    .00
        TOTAL ELAPSED TIME=     .0
000010  $HASP395 HWISTART ENDED
000200  IEA989I SLIP TRAP ID=X33E MATCHED.  JOBNAME=*UNAVAIL, ASID=012D.
000201  IEF196I           1 //IEESYSAS JOB MSGLEVEL=1
000201  IEF196I           2 //HWIBCPII EXEC IEESYSAS,PROG=HWIAMIN2
000201  IEF196I   STMT NO. MESSAGE
000201  IEF196I           2 IEFC001I PROCEDURE IEESYSAS WAS EXPANDED USING
        SYSTEM
000201  IEF196I LIBRARY SYS1.PROCLIB
000201  IEF196I           3 XXIEESYSAS PROC PROG=IEFBR14
000201  IEF196I           4 XXIEFPROC  EXEC PGM=&PROG
000201  IEF196I             XX* THE IEESYSAS PROCEDURE IS SPECIFIED IN THE
000201  IEF196I             XX* PARAMETER LIST TO IEEMB881 BY MVS COMPONENTS
000201  IEF196I             XX* STARTING FULL FUNCTION SYSTEM ADDRESS SPACES.
000201  IEF196I             IEFC653I SUBSTITUTION JCL - PGM=HWIAMIN2
000200  IEE252I MEMBER CTIHWI00 FOUND IN SYS1.IBM.PARMLIB
000201  IEF196I IEF285I    SYS1.PARMLIB                                 KEPT
000201  IEF196I IEF285I    VOL SER NOS= #@$#M1.
000201  IEF196I IEF285I    SYS1.IBM.PARMLIB                             KEPT
000201  IEF196I IEF285I    VOL SER NOS= Z1DRE1.
000010  HWI016I THE BCPII COMMUNICATION RECOVERY ENVIRONMENT IS 962
000010  NOW ESTABLISHED.
000210  HWI007I BCPII IS ATTEMPTING COMMUNICATION WITH THE LOCAL CENTRAL 963
000210  PROCESSOR COMPLEX (CPC).
000010  HWI001I BCPII IS ACTIVE.
000000  IXC104I SYSTEM STATUS DETECTION PARTITIONING PROTOCOL ELIGIBILITY: 965
          SYSTEM CANNOT TARGET OTHER SYSTEMS.
000000      REASON: SYSPLEX COUPLE DATA SET NOT FORMATTED FOR THE PROTOCOL
000000    SYSTEM IS NOT ELIGIBLE TO BE TARGETED BY OTHER SYSTEMS.
000000      REASON: SYSPLEX COUPLE DATA SET NOT FORMATTED FOR THE PROTOCOL
******************************** BOTTOM OF DATA ********************************
 COMMAND INPUT ===> _                                          SCROLL ===> CSR
  F1=HELP      F2=SPLIT      F3=END       F4=RETURN     F5=IFIND     F6=BOOK
  F7=UP        F8=DOWN       F9=SWAP nex F10=LEFT      F11=RIGHT    F12=RETRIEVE
```

What can I do with BCPii?
Off the shelf solutions

What can I do with BCPii?

- Sampling of IBM products/features using z/OS BCPii:
  - Parallel Sysplex (System Status Detect Partitioning Protocol)– V1R11 and higher
  - Capacity Provisioning Manager (CPM) – V1R10 and higher
  - Hardware Configuration Definition (HCD) – V1R13 and higher
  - Multi-Site Workload Lifeline

Why BCPii and SSDPP

- If a member of a sysplex dies, it is probably holding resources that will be required by other members of the sysplex.
- The longer this situation lasts, the more units of work will be impacted.
- The goal is to partition a dead system as quickly out of the sysplex as possible

Why BCPii and SSDPP?

Prior to z/OS 1.11, the only mechanism that z/OS had to determine the status of another member of the sysplex was to check that system's heartbeat in the sysplex CDS.

- If a system is going through recovery, it might not be able to update its heartbeat in the CDS. This means that you need to give a system some "reasonable" amount of time to recover before the system partitions the sick system out of the sysplex.
  - An IPL might take 30 minutes. Would you rather give a little more time for recovery to work, or kill it now and face an IPL? Your answer is probably "it depends on whether the system is dead or is in the middle of recovery".
  - Prior to z/OS 1.11, z/OS had no way to know whether another system was dead or trying to recover.
- SSDPP (and BCPii) changed that.

Before and after SSD (and BCPii)

- Before:
  - Wait until the Failure Detection Interval (FDI) occurs before partitioning the system out of the plex.
  - A system that wait states takes roughly just under 3 minutes before Sysplex Failure Manager (SFM) detects the problem
  - "Sympathy sickness" abounds on all systems in the sysplex for all of this time

Before and after SSD (and BCPii)

- New way:
  - Use BCPii whenever a heartbeat is missed to detect the status of the other system
  - If system is dead, SysReset that image and partition it out of the sysplex immediately
  - Systems not dead experience on average more than 2.5 minutes of relief from sympathy sickness
  - “Reply Down” message becomes mostly obsolete, reducing human error to this WTOR prompt
  - AutoIPL can be used in conjunction with this function to bring back the dead system

System Status Detection Partitioning Protocol

- **Summary:**
  - Prereqs:
    - z10 GA2 or later
    - z/OS 1.11
    - Correctly formatted Sysplex CDS
    - Implement BCPii
  - System Status Detection Partitioning Protocol is a significant step forward. This is the most fundamental change to handling of system failures since the introduction of SFM.
  - Easy to implement.
  - You can start to enable it immediately - no need to wait for the whole sysplex to be upgraded.

Capacity Provisioning Manager

- Today's challenges to manage capacity
  - ▪Unexpected events and workload spikes can afford higher processing capacity
  - ▪Manual capacity management can be time-consuming and error prone
  - ▪Capacity provisioning decisions must be made without sound data

Manual capacity upgrades – How it could look like

1.	Workload increases	0 min
2.	Operator realizes bottleneck	5-10 min
3.	Operator informs system programmers and manager	2 min
4.	Discussion	10 min
5.	Logon to HMC, change capacity	5 min

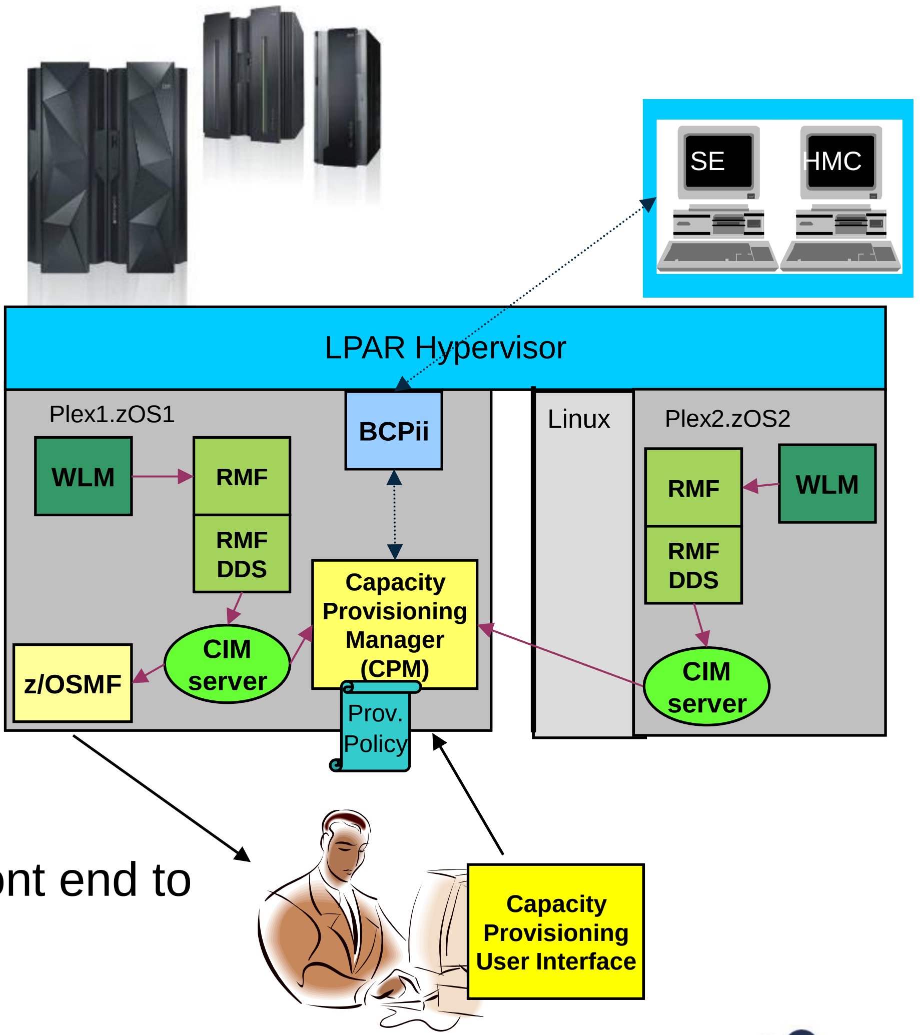
… meanwhile, so much workload may have queued up that a small amount of additional capacity would be insufficient to decrease the queued workload

→ Much more capacity has to be added

CPM can react faster and reduce cost

Capacity Provisioning – Infrastructure in a Nutshell

- z/OS WLM manages workloads to goals and business importance
- WLM indicators available through monitoring component
  - E.g. z/OS Resource Measurement Facility (RMF)
  - One RMF gatherer per z/OS system
  - RMF Distributed Data Server (DDS) per Sysplex
- Capacity Provisioning Manager (CPM) retrieves critical metrics through CIM
- CPM communicates to support elements or HMC, via BCPii.
- Capacity Provisioning User Interface is front end to administer Capacity Provisioning policies
  - z/OSMF Capacity Provisioning task



CPM Policies and Processing Parameters

- CPM server uses three types of input:
  - Domain configuration defines the topology and connections, such as the CPCs and z/OS systems that are to be managed by the server
  - Policy contains the information as to
    - which work is provisioning eligible,

      under which conditions and during which timeframes
    - how much capacity may be activated when the work suffers due to insufficient processing capacity
  - PARM data set contains setup instructions such as UNIX environment variables, and various processing options that may be set by an installation.

CPM Behavior

- When that work does not achieve its goal due to insufficient capacity and additional capacity would help, and
- The performance index of service class periods exceeds the activation threshold for a specified duration
  - Work is considered to require help and additional capacity is temporarily added using purchased capacity records
- Sophisticated controls can scope on processor limits, defined capacity limits, and group capacity limits
- Audit trails

HCD and BCPii

- HCD contains a new **S/390 Microprocessor Cluster List**, to display all logical partitions belong to the current CPC.
  - Useful information can be displayed such as:
    - Sysplex name the partition belongs
    - Name of the system running in the displayed partition
    - Operating system type
    - Operating system release level

IBM Multi-site Workload Lifeline

- Enables intelligent load balancing of TCP/IP workloads across two sites at unlimited distances to provide nearly continuous availability.
- Enables movement of workloads from one site to another by providing graceful rerouting.
- Lifeline uses BCPii to detect site failures. Lifeline communicates with all CPCs on the HMC network across both sites, to determine whether the LPARs that make up the z/OS sysplexes within each site are available. If all LPARs within a site are no longer active, then Lifeline can trigger site failure processing.
- A key component of the GDPS/Active-Active solution.

What can I do with BCPii?

- Non-IBM offerings example:
  - CA Technologies
    - CA OPS/MVS® Hardware Services utilizes BCPii to automate hardware functions such as:
      - listening for hardware events based on coding simple rules.
        - Event data is made available as simple rule variables.
      - issuing directives to the hardware easily
        - GETATTR, SETATTR and SENDCMD to specific CPC, LPAR or Activation profile entities
        - Example: can set CPU weights, CBU, COD, etc..

What can I do with BCPii?

- Non-IBM offerings example:
  - BMC Software
    - Mainview AutoOPERATOR® REXX execs utilize BCPii to List, Query, Set and issue commands to targeted hardware entities
    - “The IMFEXEC HMC capability may be interwoven with the existing automation to improve system throughput/reliability or reduce financial costs (eg. MSU based IBM charges).”
    - Examples of use: Querying and altering the GroupProfileCapacity, Changing the number of processors in an LPAR, etc..

What can I do with BCPii?

- Non-IBM offerings examples:
  - zPrice Manager®
    - Every 5 minutes zPrice Manager will compare the actual MSU usage, the 4hr MSU usage, and other parameters against rules, act accordingly and offload the information for reporting purposes. zPrice Manager accesses HMC through the BCPii interface.
      - Changes weighting, defined capacity and capacity groups based on user specified rules
  - zDynaCap® is another product from same company that also uses BCPii.

What can I do with BCPii?

- Non-IBM offerings examples:
  - zCost Management AutoSoftCapping®
    - Optimizes the performance of your system while controlling your Workload License Charges.
      - Dynamically load balances between the LPARS
      - Controlled total defined capacity
      - Web reporting
      - Forces better compliance with SLAs at the lowest cost

What can I do with BCPii?
Writing my own BCPii app

Typical BCPii usage when writing your own application

- Application to change LPAR weights at strategic times to push thru critical work
- Application to change LPAR weights to meet SLAs or to reduce costs
- Application to synchronize primary sysplex with disaster recovery site
- Remote data center operations
- React to hardware messages on the CPC
- Remote student partition administration

Programming Basics

- Services available in any address space
  - Program-authorized, and
  - SAF-authorized
- Multiple languages supported
  - C
  - Assembler
  - REXX *NEW in z/OS V2R1*
- z/OS UNIX callers can receive event notifications thru z/OS UNIX-only services utilizing the Common Event Adapter (CEA)

Programming Basics

- Interface Definition Files (IDF, or include files) provided by BCPii:
  - C *(provided in SYS1.SIEAHDRV.H)*
    - HWICIC – Main BCPii include file
    - HWIZHAPI – Additional constant definitions include file
  - Assembler *(provided in SYS1.MACLIB)*
    - HWICIASM – Main BCPii include file
    - HWIC2ASM – Additional constant definitions include file
  - REXX *(provided in SYS1.MACLIB) *NEW**
    - HWICIREX – Main BCPii include file
    - HWIC2REX – Additional constant definitions include file

Programming Basics – Samples (Non-REXX)

- BCPii sample programs *(provided in samplib)*:
  - C sample written in Metal C:
    - HWIXMCS1 provides an example of how to use all of the traditional BCPii APIs and how to construct a simple BCPii application.
    - HWIXMCX1 provides a simple example of how a BCPii Event Notification Facility (ENF) exit could be coded to field various BCPii-registered events.

Programming Basics
- New REXX Support

- **BCPii is providing a new REXX host command environment for System REXX, TSO REXX and ISV REXX environments. (address bcpii)**
  - Same authorization requirements as current BCPii applications
  - Simpler programming model than in C or Assembler
    - Programming style is intuitive for REXX programmer
    - Use of stem variables for variable number of items output
  - Parameter lists for BCPii services using REXX are simpler than C or Assembler parameter lists
    - Differences documented in the publications
  - BCPii REXX programs compatible with the different REXX environments*
  - Built-in RC return will indicate if BCPii processed the host command successfully. If zero, the BCPii return code should be consulted.

* For the common services supported by BCPii in the different environments

Programming Basics

- BCPii services available
  - **HWILIST** (BCPii List)
  - **HWICONN** (BCPii Connect)
  - **HWIDISC** (BCPii Disconnect)
  - **HWIQUERY** (BCPii Query)
  - **HWISET** (BCPii Set)
  - **HWICMD** (BCPii Command)
  - **HWIEVENT** (BCPii Event (for non-z/OS Unix callers))
  - **HwiBeginEventDelivery, HwiEndEventDelivery, HwiManageEvents, HwiGetEvent** (for z/OS Unix callers)

Programming Basics

- **HWILIST** - Retrieve HMC and BCPii configuration-related information
  - List CPCS
    - List the CPCs interconnected with the local CPC
  - List Images
    - List the images (LPARs) contained on an individual CPC or in user-defined imagegrp
  - List Capacity Records
    - List the capacity records contained on an individual CPC
  - List Events
    - List the events already registered on a particular BCPii connection
  - List Local CPC, List Local Image
    - Obtain the name of the CPC name or image (LPAR) name that the BCPii application is currently running on.
  - List Reset Activation Profiles, List Image A.P. and List Load A.P.
    - List the currently defined activation profiles contained on a individual CPC
  - List User-defined Image Group Names
    - List the currently defined image group names contained on an individual CPC.

Programming Basics

- **HWICONN** - Establish a logical connection between the application and a:
  - Central processor complex (CPC),
  - CPC image (LPAR) on a particular CPC,
  - Capacity record on particular CPC
  - Activation Profiles
  - User-defined image groups
- Input:
  - Connection type (above 3 types)
  - Connection name (CPC example: net1.cpc01)
  - Previous ConnectToken (if type is image, caprec, activation profile, or user-defined image group)
- Output:
  - ConnectToken used on subsequent BCPii calls.

Programming Basics

- **HWIDISC** – Release a logical connection no longer needed
- Input:
  - ConnectToken
- How are connections implicitly disconnected?
  - C, Assembler, System REXX
    - When a job completes associated with the BCPii application (JES or z/OS UNIX initiator), when the address space has terminated, or when the address space that invoked the System REXX exec has terminated.
  - TSO/E REXX, ISV-provided REXX environment
    - When the REXX exec ends

Programming Basics

- **HWIQUERY** - Retrieve information about objects managed by the hardware management console (HMC)/support element related to:
  - Central processor complexes (CPCs),
  - CPC images (LPARs) on a particular CPC,
  - Capacity records on particular CPC
  - Activation Profiles (Reset, Image, or Load)
  - User-defined Image group properties
- Input:
  - ConnectToken (associated with one of the above)
  - List of attributes requested, data areas to store the return values)
- Output:
  - Data returned

Programming Basics

- Examples of information you can query
  - CPC information
    - General information
      - **Name, serial, machine type, id, networking info, level of SE, engineer code & microcode levels installed**
    - Status information
      - **Operating status and other status values**
    - Capacity information
      - **Various CBU info, Capacity on Demand info, Processor configuration, including IFA, IFL, ICF, IIP**
    - Power savings information *(available on zEnterprise hardware only with APAR OA34001 on V1R10, V1R11 and V1R12)*
      - **Is power savings available?, current power savings mode, supported power saving modes available**
  - Image information
    - General information
      - **Name, OS info**
    - Capacity information
      - **Defined capacity, Processor weights**

Programming Basics

- Examples of information you can query (continued):
  - Capacity record information
    - General information
      - **Name, Activation and expiration dates, activation days**
    - Status information
      - **Record status**
    - Capacity information
      - **The entire Capacity record**
  - Activation profile information
    - Most activation profiles values

Programming Basics

- **HWISET**– Change or set data for objects managed by the hardware management console (HMC)/support element related to:
  - Central processor complexes (CPCs),
  - CPC images (LPARs) on a particular CPC,
  - Activation Profiles
- Input:
  - ConnectToken (associated with one of the above)
  - Attribute (object) to modify, the modified value, the value length
- Output
  - Return code

Programming Basics

- Examples of information you can set
  - CPC information
    - Acceptable status values
    - Next Reset activation profile name
    - Processor Running Time
  - Image information
    - Various processor weights
  - Activation Profile Information
    - Most activation profile values

Programming Basics

- **HWICMD** – Direct hardware/software commands to CPCs, images and user-defined image groups
- Input:
  - ConnectToken (associated with a CPC, image, or image group)
  - Command parameter structure (based on the type of command issued)
- Output
  - Synchronous return code
  - Asynchronous command completion event delivered to previously-registered event user when command finishes.
    - **For image commands targeted to an image group, one image event is returned for each image in the user-defined image group.**

Programming Basics

- Examples of commands that can be issued:
  - CPC commands
    - Activate, Deactivate an entire CPC
    - CBU request
      - **Activate or Undo**
    - On/Off Capacity on Demand request
      - **Activate or Undo**
    - Switch Power Savings Mode
    - Sysplex Timer (STP) commands
  - Image commands
    - SysReset, SysReset with IPL Token
    - Load
    - Start, Stop all CPs
    - Add or remove temporary capacity
    - Issue operating system command

Programming Basics

- **HWIEVENT (non-z/OS Unix callers) –** Register/Un-register an application and its connection to be notified for hardware and software events occurring on the connected CPC or image.
- Input:
  - ConnectToken (associated with a CPC or image)
  - Event action (Add or Delete)
  - Events for which an application wants to be notified
  - ENF exit to receive control when event arrives
- BCPii registers the user with ENF for this event(s) such that the ENF exit is driven only when the CPC and/or image name of the connector matches.

Programming Basics

- Examples of events that can be listened to:
  - –Command completions
  - –Status changes
  - –Capacity changes
  - –Disabled waits
  - –Power mode changes
  - –BCPii status changes and communication errors

Programming Basics
More on z/OS BCPii REXX

- z/OS BCPii APIs supported in REXX:

Services	System REXX	TSO REXX	ISV REXX
HWICONN	X	X	X
HWIDISC	X	X	X
HWILIST	X	X	X
HWIQUERY	X	X	X
HWISET	X	X	X
HWIEVENT	X		
HWICMD	X		

Programming Basics

- **z/OS BCPii System REXX support**
  - Full support of BCPii API suite
    - Command and event require non-REXX event exit and a program to wait on an ECB based on event activity
  - Ability for REXX BCPii applications to work with other C or Assembler BCPii applications
    - The Connect Token can be passed to and from the REXX exec and the other compiled BCPii applications.
  - Connections have address space affinity
    - When AXREXX macro invoker's address space terminates, BCPii will implicitly disconnect all connections
  - TSO=YES and TSO=NO environments supported
    - TSO=YES allows REXX to interpret the IBM-supplied REXX include file
    - TSO=NO requires the IBM-supplied include file to be copied into the exec
  - TIMELIMIT keyword can be used to throttle BCPii exec execution time
    - The default 30 seconds value may need to be adjusted

Programming Basics

- **z/OS BCPii System REXX support *(continued):***
  - Two methods of execution of BCPii REXX execs
    - Code an assembler program to invoke the AXREXX macro
      - Specify the name of BCPii REXX exec and any of the myriad of AXREXX options
    - New BCPii helper program HWIREXX
      - IBM-supplied helper program shipped in SYS1.LINKLIB that authorized users can invoke to launch their System REXX execs
      - Simple REXX execs can be invoked directly without the need to code the AXREXX assembler macro
      - A set of input parameters allows minor customization
        - Samplib JCL member HWIXMRJL provides list of parameters HWIREXX takes as input (supports a subset of AXREXX options)

Programming Basics

- **z/OS BCPii TSO REXX Support**
  - Support of all BCPii APIs except HWIEVENT and HWICMD
  - Connections have task affinity
    - All connections created by the REXX exec are automatically cleaned-up by BCPii when exec completes
    - Connections cannot be shared with other BCPii applications or REXX execs
  - Same SAF authorization requirements as other BCPii applications
  - Setup required for TSO REXX support
    - IKJTSOxx parmlib member must have the following update:
      - `AUTHTSF NAMES(HWIC1TRX)`

Programming Basics

- **z/OS BCPii ISV REXX Support**
  - Support of all BCPii APIs except HWIEVENT and HWICMD
  - Connections have task affinity
    - All connections created by the REXX exec are automatically cleaned-up by BCPii when exec completes
    - Connections cannot be shared with other BCPii applications or REXX execs
  - Same program and SAF authorization requirements as other BCPii applications
    - Must be invoked from an authorized address space
  - To get the “bcpii” host command environment, the REXX exec must issue the following statement:
    - `rc = hwihost("ON")`

Programming Basics

Example of z/OS BCPii REXX exec in action:

```
ListType = HWI_LIST_CPCS
address bcpii "hwilist
               ReturnCode
               ConnectToken
               ListType
               CPCList.
               DiagArea."
If rc <> 0 | ReturnCode <> 0 Then
  /* Error handling code here */
Else
  Do
    Say 'Number of CPCs returned = ' CPCList.0
    /* Write the list of CPCs returned. */
    Do i = 1 to CPCList.0
say 'CPC '|| i ' = ' CPCList.i
    End
  End
```

Programming Basics

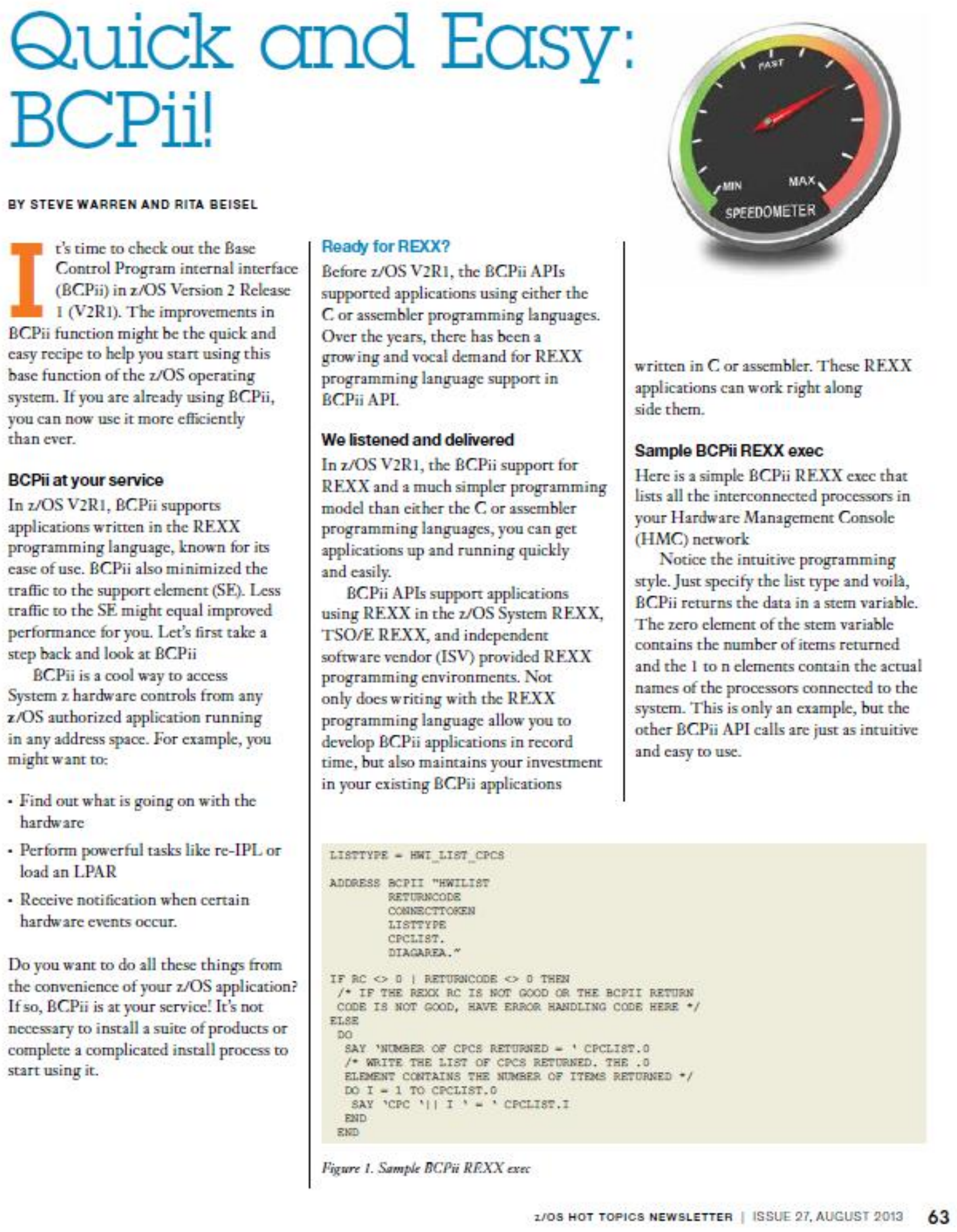
- BCPii sample programs *(provided in samplib)*:
  - REXX samples:
    - HWIXMRS1 provides a sample of how to use the connect, disconnect, list, query and set APIs in a similar format as HWIXMCS1.
    - HWIXMRS2 provide examples of using HWIEVENT and HWICMD in the System REXX environment. Assembler helper program HWIXMRA1 is required in order to run the REXX sample.
      - Sets up common storage accessible to both ENF Exit and waiting program.
      - Provides example of using the AXREXX macro to invoke the BCPii REXX exec
    - HWIXMRJL provides sample JCL to run a simple BCPii REXX under System REXX without having to code an Assembler program

BCPii Reference Materials

- **z/OS 2.1 MVS Programming: Callable Services for High-Level Languages:**
  - Primary BCPii documentation including installation instructions and BCPii API documentation (including BCPii REXX support)
- **z/OS 2.1 MVS System Commands:**
  - START HWISTART and STOP HWIBCPII commands.
- **z/OS 2.1 MVS Diagnosis: Tools and Service Aids:**
  - BCPii's CTRACE documentation.
- **z/OS 2.1 MVS Programming: Authorized Assembler Services Reference, Volume 2 (EDT-IXG):**
  - BCPii's ENF68 documentation.
- **z/OS 2.1 MVS Initialization and Tuning Reference**
  - Miscellaneous documentation
- **z/OS 2.1 MVS System Codes**
  - BCPii abend '042'x documentation

Yet More BCPii Information!

- Other SHARE presentation regarding BCPii:
  - 16090: Leveraging BCPii in Automation, presented by Zach Williams (CA)
- 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
  - z/OS Version 1 Release 13 Implementation
- 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: BCPii!

BY STEVE WARREN AND RITA BEISEL

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

BCPii at your service

In z/OS V2R1, BCPii supports applications written in the REXX programming language, known for its ease of use. BCPii also minimized the traffic to the support element (SE). Less traffic to the SE might equal improved performance for you. Let's first take a step back and look at BCPii

BCPii is a cool way to access System z hardware controls from any z/OS authorized application 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-IPL or load an LPAR
- Receive notification when certain hardware events occur.

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

Ready for REXX?

Before z/OS V2R1, the BCPii APIs supported applications using either the C or assembler programming languages. Over the years, there has been a growing and vocal demand for REXX programming language support in BCPii API.

We listened and delivered

In z/OS V2R1, the BCPii 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.

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

Sample BCPii REXX exec

Here is a simple BCPii REXX exec that lists all the interconnected processors in your Hardware Management Console (HMC) network

Notice the intuitive programming style. Just specify the list type and voilà, BCPii returns the data in a stem variable. The zero element of the stem variable contains the number of items returned and the 1 to n elements contain the actual names of the processors connected to the system. This is only an example, but the other BCPii API calls are just as intuitive and easy to use.

```
LISTTYPE = HWI_LIST_CPCS

ADDRESS BCPII "HWILIST
        RETURNCODE
        CONNECTTOKEN
        LISTTYPE
        CPCLIST.
        DIAGAREA."

IF RC <> 0 | RETURNCODE <> 0 THEN
 /* IF THE REXX RC IS NOT GOOD OR THE BCPII RETURN
 CODE IS NOT GOOD, HAVE ERROR HANDLING CODE HERE */
ELSE
 DO
  SAY 'NUMBER OF CPCS RETURNED = ' CPCLIST.0
  /* WRITE THE LIST OF CPCS RETURNED. THE .0
  ELEMENT CONTAINS THE NUMBER OF ITEMS RETURNED */
  DO I = 1 TO CPCLIST.0
   SAY 'CPC '|| I ' = ' CPCLIST.I
  END
 END
```

Figure 1. Sample BCPii REXX exec

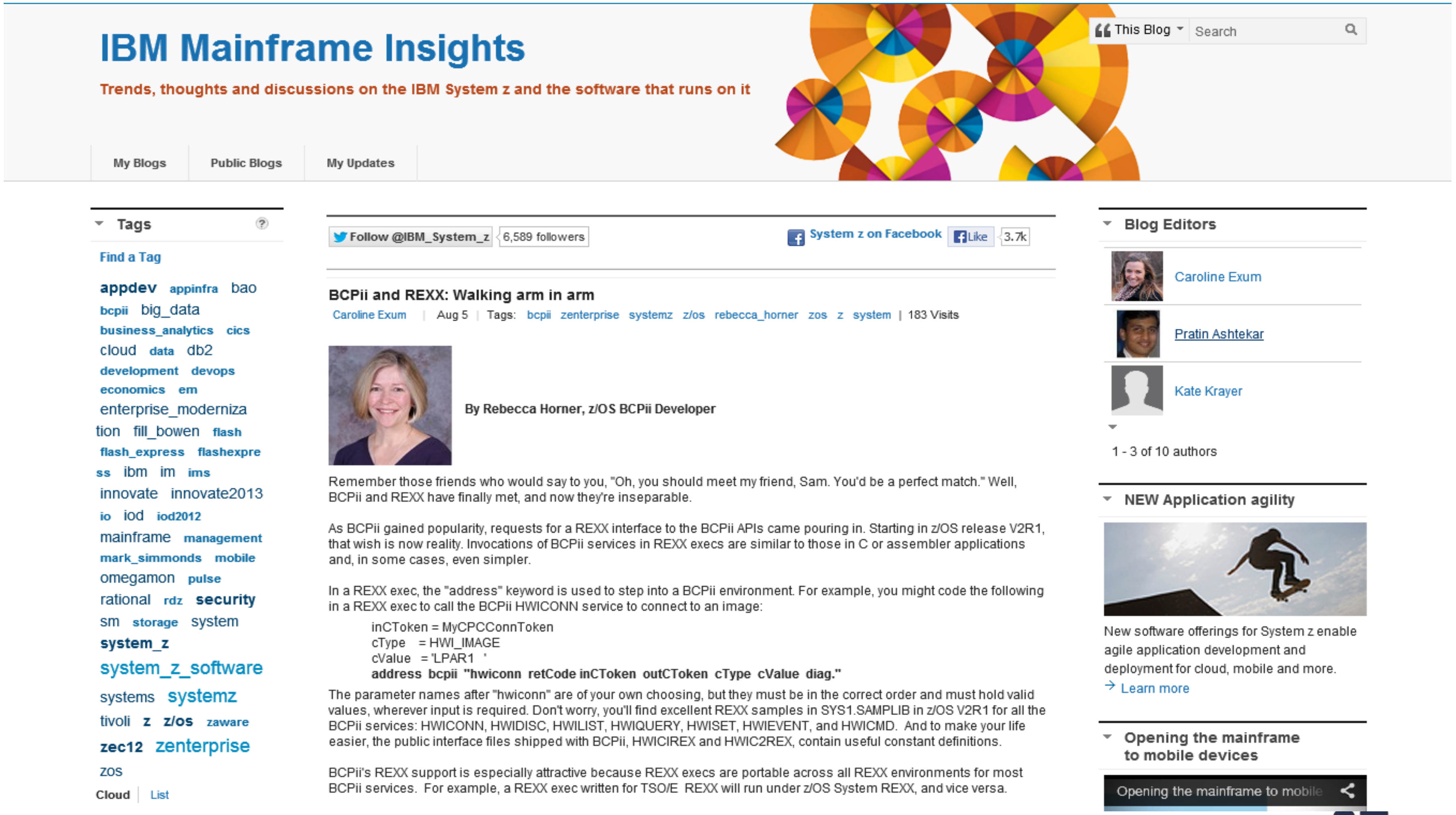
BCPii Blog

- Great new way to get tips, insight and the latest BCPii technical information
  - Hosted on IBM Mainframe Insights
  - **https://www-304.ibm.com/connections/blogs/systemz/entry/bcpii_and_rexx_walking_arm_in_arm?lang=en_us**

BCPii Blog

- Some blog entries:
  - 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

BCPii Blog Post Example



IBM Mainframe Insights

Trends, thoughts and discussions on the IBM System z and the software that runs on it

My Blogs | Public Blogs | My Updates

This Blog | Search

Tags

Find a Tag

appdev appinfra bao bcpii big_data business_analytics cics cloud data db2 development devops economics em enterprise_modernization fill_bowen flash flash_express flashexpress ibm im ims innovate innovate2013 io iod iod2012 mainframe management mark_simmonds mobile omegamon pulse rational rdz security sm storage system system_z system_z_software systems systemz tivoli z z/os zaware zec12 zenterprise zos

Cloud | List

Follow @IBM_System_z 6,589 followers | System z on Facebook Like 3.7k

BCPii and REXX: Walking arm in arm

Caroline Exum | Aug 5 | Tags: bcpii zenterprise systemz z/os rebecca_horner zos z system | 183 Visits

By Rebecca Horner, z/OS BCPii Developer

Remember those friends who would say to you, "Oh, you should meet my friend, Sam. You'd be a perfect match." Well, BCPii and REXX have finally met, and now they're inseparable.

As BCPii gained popularity, requests for a REXX interface to the BCPii APIs came pouring in. Starting in z/OS release V2R1, that wish is now reality. Invocations of BCPii services in REXX execs are similar to those in C or assembler applications and, in some cases, even simpler.

In a REXX exec, the "address" keyword is used to step into a BCPii environment. For example, you might code the following in a REXX exec to call the BCPii HWICONN service to connect to an image:

```
inCToken = MyCPCConnToken
cType    = HWI_IMAGE
cValue   = 'LPAR1   '
address bcpii "hwiconn retCode inCToken outCToken cType cValue diag."
```

The parameter names after "hwiconn" are of your own choosing, but they must be in the correct order and must hold valid values, wherever input is required. Don't worry, you'll find excellent REXX samples in SYS1.SAMPLIB in z/OS V2R1 for all the BCPii services: HWICONN, HWIDISC, HWILIST, HWIQUERY, HWISET, HWIEVENT, and HWICMD. And to make your life easier, the public interface files shipped with BCPii, HWICIREX and HWIC2REX, contain useful constant definitions.

BCPii's REXX support is especially attractive because REXX execs are portable across all REXX environments for most BCPii services. For example, a REXX exec written for TSO/E REXX will run under z/OS System REXX, and vice versa.

Blog Editors

Caroline Exum

Pratin Ashtekar

Kate Krayer

1 - 3 of 10 authors

NEW Application agility

New software offerings for System z enable agile application development and deployment for cloud, mobile and more.

Learn more

Opening the mainframe to mobile devices

Opening the mainframe to mobile

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval

Questions?

Please fill out your session evaluations!

Enjoy the rest of your week!