#SHAREorg

SHARE
Technology · Connections · Results

BCPii Programming Beyond the Basics for the z/OS System Programmer

Steve Warren

IBM

swarren@us.ibm.com

Thursday, August 9, 2012

Session Number 11713

SHARE in Anaheim 2012

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States, other countries, or both.

Not all common law marks used by IBM are listed on this page. Failure of a mark to appear does not mean that IBM does not use the mark nor does it mean that the product is not actively marketed or is not significant within its relevant market.
Those trademarks followed by ® are registered trademarks of IBM in the United States; all others are trademarks or common law marks of IBM in the United States.

For a complete list of IBM Trademarks, see www.ibm.com/legal/copytrade.shtml:

*, AS/400®, e business(logo)®, DBE, ESCO, eServer, FICON, IBM®, IBM (logo)®, iSeries®, MVS, OS/390®, pSeries®, RS/6000®, S/30, VM/ESA®, VSE/ESA, WebSphere®, xSeries®, z/OS®, zSeries®, z/VM®, System i, System i5, System p, System p5, System x, System z, System z9®, BladeCenter®

The following are trademarks or registered trademarks of other companies.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.
Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.
IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.

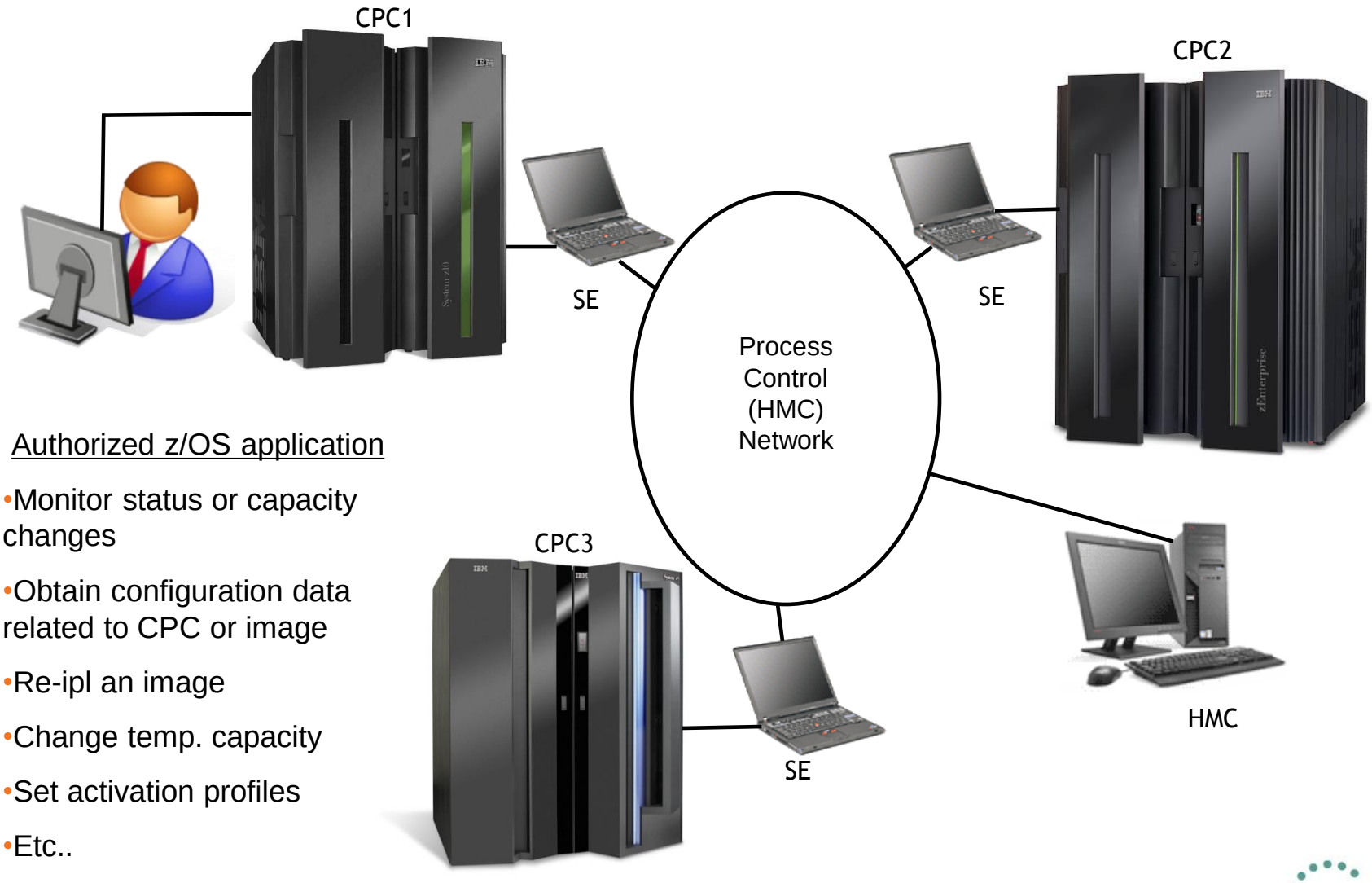
* All other products may be trademarks or registered trademarks of their respective companies.

Notes:
Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.
IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.
All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.
This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.
All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.
Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.
Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Agenda

- **BCPii quick overview**
- **BCPii Programming 101**
  - Programming Environment
  - Language support
  - Let's meet the APIs
  - Basic programming example
- **BCPii Programming 201**
  - Coding event and command
- **More advanced BCPii Programming**
  - Thinking and programming like the HMC/SE UI
  - Dynamically adapting to the current HMC config
  - Dealing with communication outages
  - Dealing with BCPii outages
  - Debugging Programming Errors

Quick 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
- Set activation profiles
- Etc..

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 - What is z/OS BCPii vs. BCPii mentioned in TSA?

- Tivoli System Automation (ProcOps) allows its automation product to use one of 2 transport protocols:
  - SNMP over an IP network
  - BCPii protocol (internal transport)
- TSA's BCPii implementation is similar but not z/OS BCPii and requires TSA, Netview and Comm Server.
- BCPii transport in TSA is for TSA usage only
- z/OS BCPii APIs can be invoked from ANY address and has no other product requirements.
- GDPS is one of the main exploiters of TSA.

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
  - Several GA'ed already
- In-house (customer-written) applications

Quick overview - BCPii Installation Steps

- Configure the local SE to support BCPii
  - HMC/SE administrator
- Authorize an application to use BCPii
  - Security administrator
- Configure the address space
  - z/OS System Administrator
- Set up the event notification mechanism for z/OS UNIX callers (if required)
  - z/OS System Administrator and Security Administrator
- See the publications or download **11806: Recent z/OS Enhancements You Can Use to Reduce Down Time** handout:
  - https://share.confex.com/share/119/webprogram/Session11806.html

Programming 101 - BCPii Execution Environment

- Hardware levels *(BCPii targeted systems)*
  - *zEnterprise*
  - *z10 plus recommended microcode levels*
    - Close to full functionality
  - *z9 plus recommended microcode levels*
    - Some reduced functionality (no IPLTOKEN, reduced attributes, no temporary capacity options)
  - *Lower than z9*
    - Significantly reduced functionality (no HWICMD, reduced attributes)
- Software levels *(System(s) which BCPii runs on)*
  - *z/OS V1R10 + PTF, z/OS V1R11 and higher in base*
  -

Programming 101 - BCPii Release Summary

- z/OS V1R10
  - Base functions (no HWISET)
- z/OS V1R11
  - HWISET
  - Support for IPL Token / Query PSWs
  - Activation profiles support
  - Minor internal serviceability enhancements
- z/OS V1R12
  - CTRACE enhancements
  - Improved storage utilization and serviceability of BCPii transport code
  - Additional CPC/Image attributes and commands

Programming 101 - BCPii Release Summary

- z/OS V1R13
  - Support for user-defined image groups
  - Additional CPC/Image attributes
  - New STP commands

Programming 101 - Programming Environment

- Services available in any address space
  - Program-authorized, and
  - SAF-authorized
- C and Assembler programming languages
  - REXX: FITS requirements MR0409106649 and MR0930096444 asking for BCPii System REXX support answered as “Accepted” by IBM on 1/24/11.
- z/OS UNIX callers can receive event notifications thru z/OS UNIX-only services utilizing the Common Event Adapter (CEA)

Programming 101 – Language Support

- 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

Programming 101 – Programming Environment

- Two ways to link your BCPii program:
  - Use the linkable stub routine HWICSS from SYS1.CSSLIB to link-edit your object code.
  - Use the LOAD macro to find the address of the BCPii callable service at run time and then CALL the service

Programming 101 - Programming Environment - Samples

- 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 101 - Let's Meet the APIs!

- Functions performed using BCPii APIs:
  - Obtain the System z topology of the current interconnected CPCs, Images (LPARs) and their associated capacity records, activations profiles and user-defined image groups
  - Query various CPC, image (LPAR), capacity record, activation profile and user-define image group information
  - Set various CPC, image(LPAR), and activation profile information
  - Issue commands against both the CPC and image to perform hardware and software-related functions
  - Listen for various hardware and software events which may take place on various CPC and images

Programming 101 - Let's Meet the APIs!

- Services available
  - HWILIST (BCPii List)
  - HWICONN (BCPii Connect)
  - HWIDISC (BCPii Disconnect)
  - HWIQUERY (BCPii Query)
  - HWISET (BCPii Set) – introduced in V1R11
  - HWICMD (BCPii Command)
  - HWIEVENT (BCPii Event (for non-z/OS Unix callers))
  - HwiBeginEventDelivery, HwiEndEventDelivery, HwiManageEvents, HwiGetEvent (for z/OS Unix callers)

Programming 101 - Let's Meet the APIs!

- Services available
  - HWILIST (BCPii List)
  - HWICONN (BCPii Connect)
  - HWIDISC (BCPii Disconnect)
  - HWIQUERY (BCPii Query)
  - HWISET (BCPii Set) – introduced in V1R11
  - HWICMD (BCPii Command)
  - HWIEVENT (BCPii Event (for non-z/OS Unix callers))
  - HwiBeginEventDelivery, HwiEndEventDelivery, HwiManageEvents, HwiGetEvent (for z/OS Unix callers)

Programming 101 - Let's Meet the APIs!

- All services pass back two groups of information used to determine the success of the request
  - Return code
    - 0 – Request completed successfully
  - DiagArea
    - Area that is filled in for certain non-environmental failures

Field Name	Field Type	Description
Diag_Index	32-bit integer	The array index to the parameter field that causes the error.
Diag_Key	32-bit integer	The constant value represents the field that causes the error.
Diag_Actual	32-bit integer	The incorrect actual value specified.
Diag_Expected	32-bit integer	The expected value to be used.
Diag_CommErr	32-bit integer	The returned code that is returned from the console application API or the BCPii transport layer.
Diag_Text	Character (12)	Additional diagnostic information in text format.

Programming 101 - Let's Meet the APIs!

- **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 (available in V1R11)
    - 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. *(available in V1R11 via APAR OA29638)*
    - 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 101 - Let's Meet the APIs!

```
CALL HWILIST (
 ReturnCode
,ConnectToken
,ListType
,NumofDataItemsReturned
,AnswerArea_Ptr
,AnswerAreaLen
,DiagArea)
```

Programming 101 - Let's Meet the APIs!

- **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 101 - Let's Meet the APIs!

```
CALL HWICONN (
 ReturnCode
,InConnectToken
,OutConnectToken
,ConnectType
,ConnectTypeValue_Ptr
,DiagArea)
```

Programming 101 - Let's Meet the APIs!

- **HWIDISC** – Release a logical connection no longer needed
- Input:
  - ConnectToken
- Note: Connections are implicitly disconnected when a job completes associated with the BCPii application (JES or z/OS UNIX initiator) or when the address space has terminated

Programming 101 - Let's Meet the APIs!

```
CALL HWIDISC  (
 ReturnCode
,ConnectToken
,DiagArea)
```

Programming 101 - Let's Meet the APIs!

- **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 101 - Let's Meet the APIs!

- Examples of information you can query
  - CPC information
    - General information
      - **Name, serial, machine type, id, networking info**
    - 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 101 - Let's Meet the APIs!

- 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 101 - Let's Meet the APIs!

```
CALL HWIQUERY  (
 ReturnCode
,ConnectToken
,QueryParm_Ptr
,NumOfAttributes
,DiagArea)
```

Structure of one QueryParm:

Field Name	Field Type
AttributeIdentifier	32-bit unsigned integer
AttributeValue_Ptr	Pointer
AttributeValueLen	32-bit unsigned integer
AttributeValueLenReturned	32-bit unsigned integer

Programming 101 - Let's Meet the APIs!

- HWISET *(available in V1R11)* – 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 *(available in V1R11 via APAR OA29638)*
- Input:
  - ConnectToken (associated with one of the above)
  - Attribute (object) to modify, the modified value, the value length
- Output
  - Return code

Programming 101 - Let's Meet the APIs!

```
CALL HWISET  (
 ReturnCode
,ConnectToken
,SetType
,SetTypeValue_Ptr
,SetTypeValueLen
,DiagArea)
```

Programming 101 - Let's Meet the APIs!

- 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 *(available in V1R11 via APAR OA29638)*
    - Most activation profile values

Programming 101 - Let's Meet the APIs!

- **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 101 - Let's Meet the APIs!

- 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 *(available on zEnterprise hardware only with APAR OA34001 on V1R10, V1R11 and V1R12)*
  - Image commands
    - SysReset, SysReset with IPL Token *(V1R11)*
    - Load
    - Start, Stop all CPs
    - Add or remove temporary capacity
    - Issue operating system command

Programming 101 - Let's Meet the APIs!

```
CALL HWICMD  (
 ReturnCode
,ConnectToken
,CmdType
,CmdParm_Ptr
,DiagArea)
```

CmdParm_Ptr points to the command parameter list that is unique for each command. The data structure specified is defined in the IBM-supplied IDF files.

Programming 101 - Let's Meet the APIs!

- **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 101 - Let's Meet the APIs!

- Examples of events that can be listened to:
  - Command completions
  - Status changes
  - Capacity changes
  - Disabled waits
  - Power mode changes *(available on zEnterprise hardware only with APAR OA34001 on V1R10, V1R11 and V1R12)*
  - BCPii status changes and communication errors

Programming 101 - Let's Meet the APIs!

```
CALL HWIEVENT  (
 ReturnCode
,EventAction
,EventIDs
,EventExitMode
,EventExitAddr
,EventExitParm
,DiagArea)
```

Note: EventIDs is a 128 bit data structure is defined in the IBM-supplied IDF files as HWI_EVENTIDS_TYPE. Specify which events you wish to register with by turning on the appropriate bits. Make sure to fill in the beginning "eyecatcher" field with the constant value "HWIEVENTBLCK"

Programming 101 - Let's Meet the APIs!

- **HwiBeginEventDelivery (z/OS Unix callers)** – begin delivery of event notifications.
- Input:
  - ConnectToken (associated with a CPC or image)
- Output:
  - DeliveryToken
    - To be used on HwiManageEvents service

Programming 101 - Let's Meet the APIs!

- **HwiEndEventDelivery (z/OS Unix callers)** – End delivery of event notifications.
- Input:
  - DeliveryToken

Programming 101 - Let's Meet the APIs!

- **HwiManageEvents (z/OS Unix callers)** – Registers / un-registers for a list of hardware/software events.
- Input:
  - ConnectToken
  - DeliveryToken
  - Event action (Add or Delete)
  - Events to be registered/unregistered

Programming 101 - Let's Meet the APIs!

- **HwiGetEvent (z/OS Unix callers)** – Retrieve outstanding event notifications.
- Input:
  - DeliveryToken
  - Buffer
    - Where the ENF68 event data is to be returned
  - Timeout
    - How much time to wait for an event to occur
- Output:
  - ENF68 Event Data in supplied buffer

Programming 101 - Simple Programming Example

- Application contains calls like this:
  - **HWILIST** (ListCPCs)
  - For each CPC name returned above:
    - **HWICONN** (CPC name (input), CPCConnectToken(output))
    - **HWIQUERY** (CPCConnectToken (input), QueryParms (HWI_CBUTESTAR))
    - **HWILIST** (CPCConnectToken(input), ListImages)
    - For each image returned above:
      - **HWICONN** (CPCConnectToken(input), Image name (input), ImageConnectToken(output))
      - **HWIQUERY** (ImageConnectToken(input), QueryParms(HWI_OSNAME,HWI_OSTYPE,HWI_OSLEVEL,HWI_SYSPLEX,HWI_DEFCAP)
      - **HWIDISC**(ImageConnectToken)
    - **HWIDISC** (CPCConnectToken)

Programming 101 - Simple Programming Example (HWILIST)

```
rc = -1;
numofCPCs = -1;
listtype = HWI_LIST_CPCS;
memset(List_of_CPCs, 0x00, sizeof(List_of_CPCs));
answerarealen = sizeof(List_of_CPCs);
answerarea_ptr = &answerarea[0];

hwilist(&rc,CPCoutconnecttoken,listtype,&numofCPCs,
        &answerarea_ptr,answerarealen,&diagarea);

if ( rc == 0 )
{
 printf("HWILIST for CPC: RC = %x\n",rc);
 printf("NumOfDataItem:   %d\n",numofCPCs);

 CPC_AnswerArea_into_Array(answerarea, List_of_CPCs, numofCPCs);
```

Programming 101 - Simple Programming Example (HWICONN)

```
while( i < numofCPCs )
{
 printf("CPC %d: %s\n",i+1,&List_of_CPCs[i].element);
 /* CPC ***************************************************/
 /* HWICONN the CPC */
 rc = -1;
 memset(CPCinconnecttoken, 0x00, sizeof(CPCinconnecttoken));
 strcpy(CPC_target,List_of_CPCs[i].element);
 CPCconnecttypevalue = &CPC_target[0];
 CPCconnecttype = 1;

 hwiconn(&rc,CPCinconnecttoken,&CPCoutconnecttoken,CPCconnecttype,
         &CPCconnecttypevalue,&diagarea);

 if ( rc == 0 )
 {
  printf("HWICONN on %s: RC = %x\n",&List_of_CPCs[i],rc);
```

Programming 101 - Simple Programming Example (HWIQUERY)

```
/* HWIQUERY */
/* Calling HWIQUERY, using the returned output connecttoken from
HWICONN, query for HWI_MMODEL */
printf("HWIQUERY for HWI_MMODEL\n");
rc = -1;
numofattributes = 1;
Queryparm[0].AttributeIdentifier = HWI_MMODEL;
Queryparm[0].AttributeValue_Ptr  = &HWI_MMODEL_value[0];
Queryparm[0].AttributeValueLen   = sizeof(HWI_MMODEL_value);
Queryparm[0].AttributeValueLenReturned = -1;
query_ptr = (char *)&queryparm[0];

hwiquery(&rc,CPCoutconnecttoken,(void **)&query_ptr
         ,numofattributes,&diagarea);

if ( rc == 0 )
{printf("HWIQUERY on %s: RC = %x\n",&List_of_CPCs[i],rc);
```

Programming 201 - Programming Example (HWIEVENT)

- Events are driven in a BCPii thread as they occur
  - ENF exit is driven
  - ENF exit needs to wake up user's mainline program to perform some sort of action based on the event that was driven.
  - Posting an ECB waited on by the mainline application is easy.
  - Obtain a small piece of common storage for the ECB, and pass the address of the ECB on the HWIEVENT call.
  - Mainline program waits on the ECB
  - When the ENF is driven, and the ENF event exit needs to wake up the main program, a simple Post instruction does the trick.

Programming 201 - Programming Example (HWIEVENT)

```
memcpy(&eventExitParm,&userdata,sizeof(eventExitParm));
memset(&eventIDs,0,sizeof(eventIDs));
strcpy(eventIDs.Hwi_EventID_EyeCatcher,HWI_EVENTID_TEXT);
eventIDs.Hwi_Event_CmdResp = 1;
eventIDs.Hwi_Event_DisabledWait = 1;
eventAction = HWI_EVENT_ADD;
eventExitMode = HWI_EVENT_TASK;
eventExitAddr = pECBandCtoken;
__asm ( " LOAD EP=HWIXMCX1  " : "=r"(eventExitEP) :  );
eventExitAddr = (int)eventExitEP;
/* -------------- */
/* Call HWIEVENT  */
/* ---------------*/
hwievent(returncodePtr,
         connecttoken,
         eventAction,
         eventIDs,
         eventExitMode,
         eventExitAddr,
         &eventExitParm,
         &diagarea);
```

Programming 201 - Programming Example (HWICMD)

```
/* ------------------------------------------------------------*/
/* Initialize the cmdParm to null.                             */
/* Note: An OS command string must be null-terminated.         */
/* ------------------------------------------------------------*/
memset(&cmdParm,0,sizeof(cmdParm));
cmdParm_Ptr = &cmdParm;
cmdParm.PriorityType = HWI_CMD_NONPRIORITY;
strcpy(cmdParm.OSCMDString,"D GRS");
cmdType = HWI_CMD_OSCMD;
HWI_DIAGAREA_TYPE  diagarea;
/* ------------ */
/* Call HWICMD  */
/* ------------ */
hwicmd(returncodePtr,
       connecttoken,
       cmdType,
       &cmdParm_Ptr,
       &diagarea);

if(*returncodePtr) print_diagarea(diagarea);
```

Programming 201 - Coding the ENF Event Exit

- BCPii uses ENF 68
- 3 Types of BCPii ENF Signals
  - **HWIENF68_EVENTTYPE_BCPIISTATUS**
    - HWIENF68_BCPIISTATUS_AVAIL
    - HWIENF68_BCPIISTATUS_UNAVAIL
  - **HWIENF68_EVENTTYPE_HWCOMMERROR**
    - HWIENF68_HWCOMMERROR_TEMP
    - HWIENF68_HWCOMMERROR_PERM
    - HWIENF68_HWCOMMERROR_AVAIL
  - **HWIENF68_EVENTTYPE_HWEVENT**

Programming 201 - Coding the ENF Event Exit

- **HWIENF68_EVENTTYPE_HWEVENT Subtypes**
  - HWIENF68_HWEVENT_CMDRESP
  - HWIENF68_HWEVENT_STATUSCHG
  - HWIENF68_HWEVENT_NAMECHG
  - HWIENF68_HWEVENT_ACTPROFCHG
  - HWIENF68_HWEVENT_OBJCREATE
  - HWIENF68_HWEVENT_OBJDESTROY
  - HWIENF68_HWEVENT_OBJEXCEPTION
  - HWIENF68_HWEVENT_APPLSTARTED
  - HWIENF68_HWEVENT_APPLENDED
  - HWIENF68_HWEVENT_OPSYSMSG
  - HWIENF68_HWEVENT_HWMSG
  - HWIENF68_HWEVENT_HWMSGDEL
  - HWIENF68_HWEVENT_CAPACITYCHG
  - HWIENF68_HWEVENT_CAPACITYRECORD
  - HWIENF68_HWEVENT_SECURITYEVENT
  - HWIENF68_HWEVENT_DISABLEDWAIT
  - HWIENF68_HWEVENT_POWERCHANGE

Programming 201 - Coding the ENF Event Exit

- Each hardware event contains unique data specific to that event.
  - Each unique hardware event has its own data mapping
  - Example: HWIENF68_HWEVENT_CMDRESP

```
typedef struct ??<                    /* Command response              */
  HWI_CONNTOKEN_TYPE connectToken;    /* Connect Token            @03A */
  HWIENF68_STRING_T eventObjName;     /* Affected object name          */
  HWIENF68_INT_T    cmdType;          /* Type of command issued        */
  HWIENF68_INT_T    cmdRetCode;       /* Command return code           */
  HWIENF68_BOOL_T   lastResponse;     /* If true, the last response    */

??> HWIENF68_CMDRESP_T;
```

Programming 201 - Programming Example (Event Exit)

```
typedef struct {
  HWIENF68 * ENFEventDataPtr;   /* Data for a specific BCPii event   */
  int    reserved1;
  int    ENFUserData;           /* Optional user-supplied data       */
  int    reserved2;
  int    reserved4;
  int    reserved5;
} ENFDATA_TYPE;

 int main(ENFDATA_TYPE ENFData)
{
switch (ENFData.ENFEventDataPtr->eventType)
  {
```

Programming 201 - Programming Example (Event Exit)

```
/* ------------------------------------------------------------ */
   /* Event Type Hardware Event                                  */
   /* ---------------------------------------------------------- */
   case HWIENF68_EVENTTYPE_HWEVENT:
     /* ------------------------------------------------------- */
     /* Check for a specific event subtype                      */
     /* ------------------------------------------------------- */
     switch (ENFData.ENFEventDataPtr->eventSubType)
       {
       /* ----------------------------------------------------- */
       /* Handle Command Response event subtype.                */
       /* ----------------------------------------------------- */
       case HWIENF68_HWEVENT_CMDRESP:
```

Programming 201 - Programming Example (Event Exit)

```
          /* ------------------------------------------------------- */
          /* Look at the eventdata fields for this eventSubType      */
          /* which are mapped by the HWIENF68_CMDRESP_T              */
          /* structure.                                              */
          /* ------------------------------------------------------- */
          /* Check whether this command response is one for          */
          /* which you are waiting by comparing the connect          */
          /* token to a saved connect token.                         */
*/
          /* ------------------------------------------------------- */
          /* Validate a saved connect token.                         */
          /* -------------------------------------------------------
  {if(savedConnectToken == ENFData.ENFEventDataPtr->
                        eventData.CmdResp.connectToken)
*/
if(1)                           /* If connect token matches */
   . . . . . .
```

More Advanced BCPii Programming – Thinking and programming like the HMC/SE UI

- The HMC/SE has dependencies between many attributes found on the screen. Here are some examples:
  - WLM checkbox in image and image activation profiles (means Work Load Manager is/is not allowed to change processing weight-related attributes.)
    - If checked, then certain attributes cannot be set on or consulted.
    - Initial processing weight capped (Hwi_SGPIPWCAP) and other attributes cannot be turned on or consulted if Hwi_WLM is on
    - Defined capacity (Hwi_Defcap) cannot be consulted if Hwi_WLM is turned off

More Advanced BCPii Programming – Thinking and programming like the HMC/SE UI

EBITHMC1: Customize/Delete Activation Profiles - Mozilla Firefox: IBM Edition

9.12.16.166 https://9.12.16.166/hmc/content?taskId=677&refresh=951

Customize Image Profiles: R132:LP3 : LP3 : Processor

- R132:LP3
  - LP3
    - General
    - Processor
    - Security
    - Storage
    - Options
    - Load
    - Crypto

Group Name <Not Assigned>

Logical Processor Assignments

Dedicated processors

Select	Processor Type	Initial	Reserved
☑	Central processors (CPs)	3	0
☐	zSeries application assist processors (zAAPs)	0	0
☐	System z integrated information processors (zIIPs)	0	0

Not Dedicated Processor Details

Initial processing weight 10 1 to 999 Initial capping

☑ Enable workload manager

Minimum processing weight 0

Maximum processing weight 0

Cancel | Save | Copy Profile | Paste Profile | Assign Profile | Help

- WLM checkbox checked
- Minimum processing weight and maximum processing weight available to be set.

More Advanced BCPii Programming – Thinking and programming like the HMC/SE UI

EBITHMC1: Customize/Delete Activation Profiles - Mozilla Firefox: IBM Edition

9.12.16.166 https://9.12.16.166/hmc/content?taskId=677&refresh=951

Customize Image Profiles: R132:LP3 : LP3 : Processor

- R132:LP3
  - LP3
    - General
    - Processor
    - Security
    - Storage
    - Options
    - Load
    - Crypto

Group Name <Not Assigned>

Logical Processor Assignments

Dedicated processors

Select	Processor Type	Initial	Reserved
☑	Central processors (CPs)	3	0
☐	zSeries application assist processors (zAAPs)	0	0
☐	System z integrated information processors (zIIPs)	0	0

Not Dedicated Processor Details

Initial processing weight 10 1 to 999 Initial capping

Enable workload manager

Minimum processing weight 0

Maximum processing weight 0

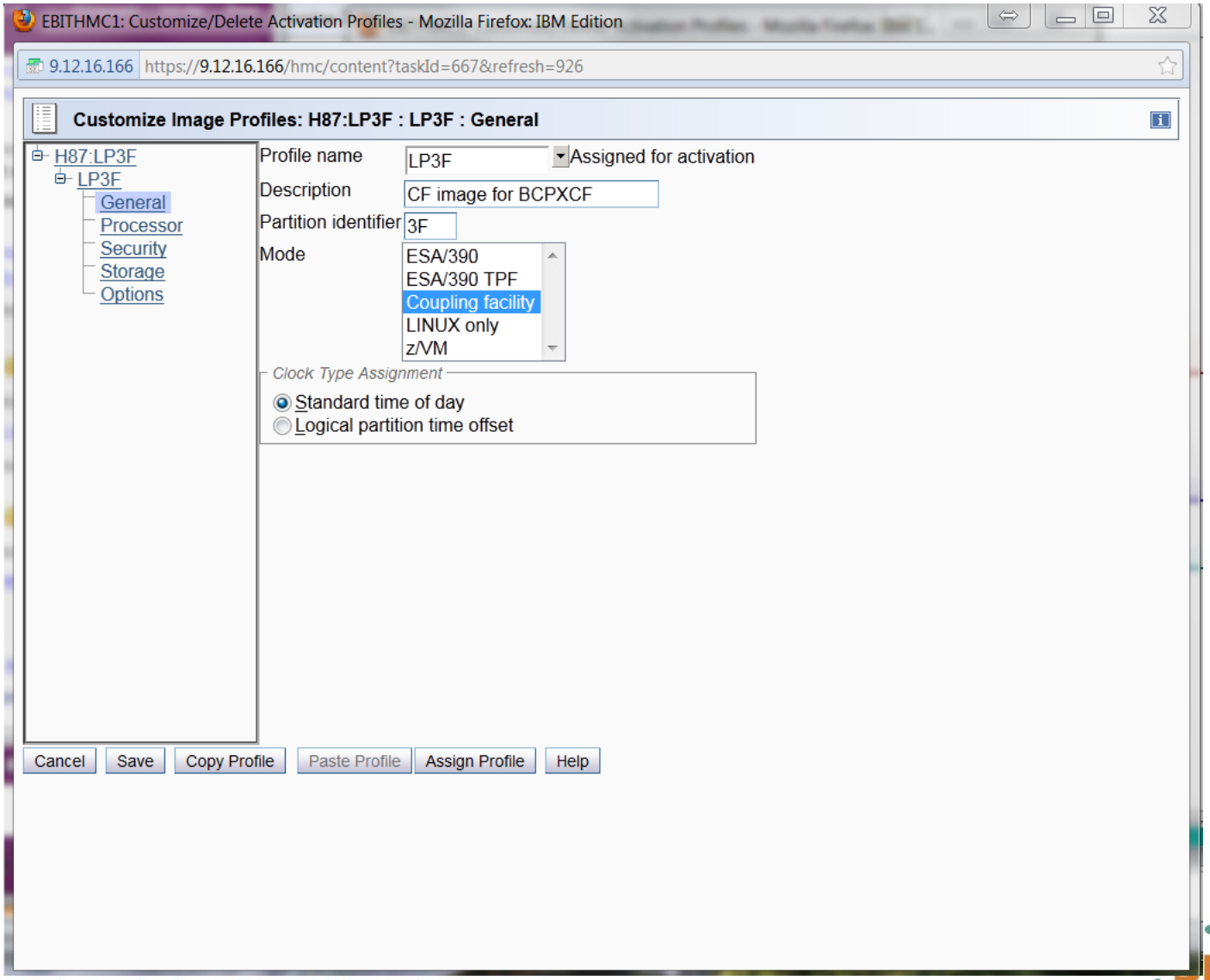
Cancel | Save | Copy Profile | Paste Profile | Assign Profile | Help

- WLM checkbox unchecked
- Minimum processing weight and maximum processing weight grayed out. Values are not settable.

More Advanced BCPii Programming – Thinking and programming like the HMC/SE UI

- Activation profiles defined to be in a certain mode can only have to certain specialty engines
  - Example: Don't consult CF attributes of the image activation profile if no ICF specialty engines are made available to the image activation profile, regardless of the number of ICF engines installed on the CPC (HWI_NUMICFP).

More Advanced BCPii Programming – Thinking and programming like the HMC/SE UI



More Advanced BCPii Programming – Thinking and programming like the HMC/SE UI

EBITHMC1: Customize/Delete Activation Profiles - Mozilla Firefox: IBM Edition

9.12.16.166 https://9.12.16.166/hmc/content?taskId=675&refresh=942

Customize Image Profiles: H87:LP3F : LP3F : Processor

- H87:LP3F
  - LP3F
    - General
    - Processor
    - Security
    - Storage
    - Options

Group Name <Not Assigned>

Logical Processor Assignment

- Dedicated central processors
- Dedicated internal coupling facility processors
- Not dedicated central processors
- Not dedicated internal coupling facility processors
- Dedicated internal coupling facility processors and not dedicated central processors
- Dedicated and not dedicated internal coupling facility processors

Not Dedicated Processor Details

Initial processing weight 10 1 to 999 Initial capping

(Maximum of 36 initial central processors and/or 16 initial internal coupling facility processors.)
(Maximum of 56 initial and reserved central processors and/or 16 initial and reserved internal coupling facility processors.)

Number of processors - Initial 2 Reserved 0

Cancel | Save | Copy Profile | Paste Profile | Assign Profile | Help

More Advanced BCPii Programming – Thinking and programming like the HMC/SE UI

- Find the number of ICF processors available for this image activation profile:

```
int Num_ICF, Num_ResICF, Num_SharedICF, Num_ResSharedICF;
#define NUMOFACTPROFATTRIBUTES 4
HWI_QUERYPARM_TYPE queryparm[NUMOFACTPROFATTRIBUTES];
queryparm[0].AttributeIdentifier = HWI_NUM_ICF;
queryparm[0].AttributeValue_Ptr  = (char *)&Num_ICF;
queryparm[0].AttributeValueLen=sizeof(int);
queryparm[0].AttributeValueLenReturned = 0;

queryparm[1].AttributeIdentifier = HWI_NUM_RESICF;
queryparm[1].AttributeValue_Ptr  = (char *)&Num_ResICF;
queryparm[1].AttributeValueLen=sizeof(int);
queryparm[1].AttributeValueLenReturned = 0;

queryparm[2].AttributeIdentifier = HWI_NUM_SHARED_ICF;
queryparm[2].AttributeValue_Ptr  = (char *)&Num_SharedICF;
queryparm[2].AttributeValueLen=sizeof(int);
queryparm[2].AttributeValueLenReturned = 0;

queryparm[3].AttributeIdentifier = HWI_NUM_RES_SHARED_ICF;
queryparm[3].AttributeValue_Ptr  = (char *)&Num_ResSharedICF;
queryparm[3].AttributeValueLen=sizeof(int);
queryparm[3].AttributeValueLenReturned = 0;
```

More Advanced BCPii Programming – Thinking and programming like the HMC/SE UI

```
/* ------------- */
/* Call HWIQUERY */
/* ------------- */
queryparm_Ptr = (char *)&queryparm[0];
hwiquery(returncodePtr,
         connecttoken,
         &queryparm_Ptr,
         NUMOFIMAGEATTRIBUTES,
         &diagarea);

if ( *returncodePtr == 0 )
  {
   if ((Num_ICF !=0) |
       (Num_ResICF !=0) |
       (Num_SharedICF !=0) |
       (Num_ResSharedICF !=0))
      {
       /* Possible to query the HWI_ICFIPW, HWI_ICFIPWCAP, HWI_ICFPWMIN,
       HWI_ICFPWMAX, HWI_ICFPW, HWI_ICFPWCAP attributes for this actprof. */
      }
   }
```

More Advanced BCPii Programming – Thinking and programming like the HMC/SE UI

> ICFs available to the image in this profile: 2

> Reserved ICFs available to the image in this profile: 0

> Shared ICFs available to the image in this profile: 0

> Reserved Shared ICFs available to the image in this profile: 0

- Profile has coupling facility engines available to it
- ICF attributes can be consulted for this activation profile.
- If the numbers were all zero, trying to consult an ICF attribute would result in an error return code from BCPii.

More Advanced BCPii Programming – Dynamic Config Handling

- If your program is long running, new entities may be added to the system, old ones deleted, or names changed. For example, activation profiles.
- Consider registering for HWIENF68_HWEVENT_NAMECHG, HWIENF68_HWEVENT_OBJCREATE, and HWIENF68_HWEVENT_OBJDESTROY events
- When activation profile is added, for example, the OBJCREATE event will be driven with the name of the object, the object type (e.g. Image actprof) if the version number is greater than 1 (datavers field in the HWIENF68), and the name of the CPC (cpcName field in the HWIENF68)
- Your program takes the appropriate action.
- Note: Please apply APAR OA38252.

More Advanced BCPii Programming– Handling Communication Outages

- It is possible that BCPii could lose connectivity to the CPC that you are connected to and waiting for events on.
- BCPii can tell you about these outages, allowing your program to take the appropriate actions:
  - HWIENF68_HWCOMMERROR_TEMP (ENF Qual 02010001)
    - BCPii detected that it lost connectivity momentarily with the target CPC but has regained connectivity
    - BCPii application should take the appropriate action
  - HWIENF68_HWCOMMERROR_PERM (ENF Qual 02010002)
    - BCPii detected that it lost connectivity and cannot regain connectivity to the CPC at the moment.
    - All HWIEVENT and HWICMD API requests are not processed, no events are delivered
    - BCPii application should wait for the HWIENF68_HWCOMMERROR_AVAIL event

More Advanced BCPii Programming– Handling Network Outages

- HWIENF68_HWCOMMERROR_AVAIL (ENF Qual 02010003)
  - BCPii has established communications with the registered CPC (either at first connectivity to that CPC or when communications have resumed to that CPC

- An application has a choice of how to register for all hardware communication events:
  - Via HWIEVENT ADD service (EventIDs parameter value Hwi_Event_HwCommError)
  - Via ENFREQ LISTEN macro invocation specifying to listen for ENF68,
    - ENFREQ ACTION=LISTEN
    - CODE=ENFPC068, QUAL=02010000

More Advanced BCPii Programming– Handling Communication Outages

```
case HWIENF68_EVENTTYPE_HWCOMMERROR:
  switch (ENFData.ENFEventDataPtr->eventSubType)
    {
     case HWIENF68_HWCOMMERROR_TEMP:
       /* Take appropriate actions.                                  */
       break;
     case HWIENF68_HWCOMMERROR_PERM:
       /* Take appropriate actions.                                  */
       break;
     case HWIENF68_HWCOMMERROR_AVAIL:
       /* Take appropriate actions.                                  */
       break;
     default:
       /* Unknown BCPii Communication Error value returned.          */
       /* Take appropriate actions.                                  */
       break;
    } /* end switch on the value of the BCPii Communication error evt */
```

More Advanced BCPii Programming– Handling BCPii Outages

- While very rare, it is possible for the BCPii address space to go away unexpectedly.
  - BCPii signals an ENF68 with a QUAL of 01000002 when the address space becomes active
  - BCPii signals an ENF68 with a QUAL of 01000001 when the address space becomes unavailable
- An application should register itself with ENF to listen for these two events from occurring so it can take the appropriate actions
  - When BCPii goes down, all connections are lost. The application should throw away all recollection of connect tokens
  - When BCPii come back up, all connections should be reestablished.

More Advanced BCPii Programming – Debugging Programming Errors

- API Return Codes and Diag Area
- CTRACE
  - BCPii cuts CTRACE records using SYSBCPII CTRACE comp
  - Default CTRACE CTIHWI00 parmlib member shipped
  - Two CTRACE options:
    - Min
    - All
  - Dump is taken whenever CTRACE is turned off
- Symptom Records
  - Limited first failure data capture for select problems
- Support Element Tracing

BCPii Publications

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

Other BCPii information

- Other SHARE presentations given this week regarding BCPii:
  - **11806: Recent z/OS Enhancements You Can Use to Reduce Down Time**, presented by Frank Kyne and Karan Singh.
  - **12088: (Hardware Management Console) Security Basics & Best Practices,** presented by Brian Valentine
- IBM Redbooks (http://www.redbooks.ibm.com)
  - **System z Parallel Sysplex Best Practices**
  - **z/OS Version 1 Release 13 Implementation**
- Other publications
  - z/OS Hot Topics
    - **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)***

Questions?

- Please fill out the online session evaluation at either:
  - SHARE.org/AnaheimEval, or
  - Aim your smartphone at this QR code below: