#SHAREorg

What's New in BCPii in z/OS 2.1? Full REXX Support and Faster Data Retrieval

Steve Warren

swarren@us.ibm.com

SHARE Boston

Session 13836

August 13, 2013

SHARE in Boston

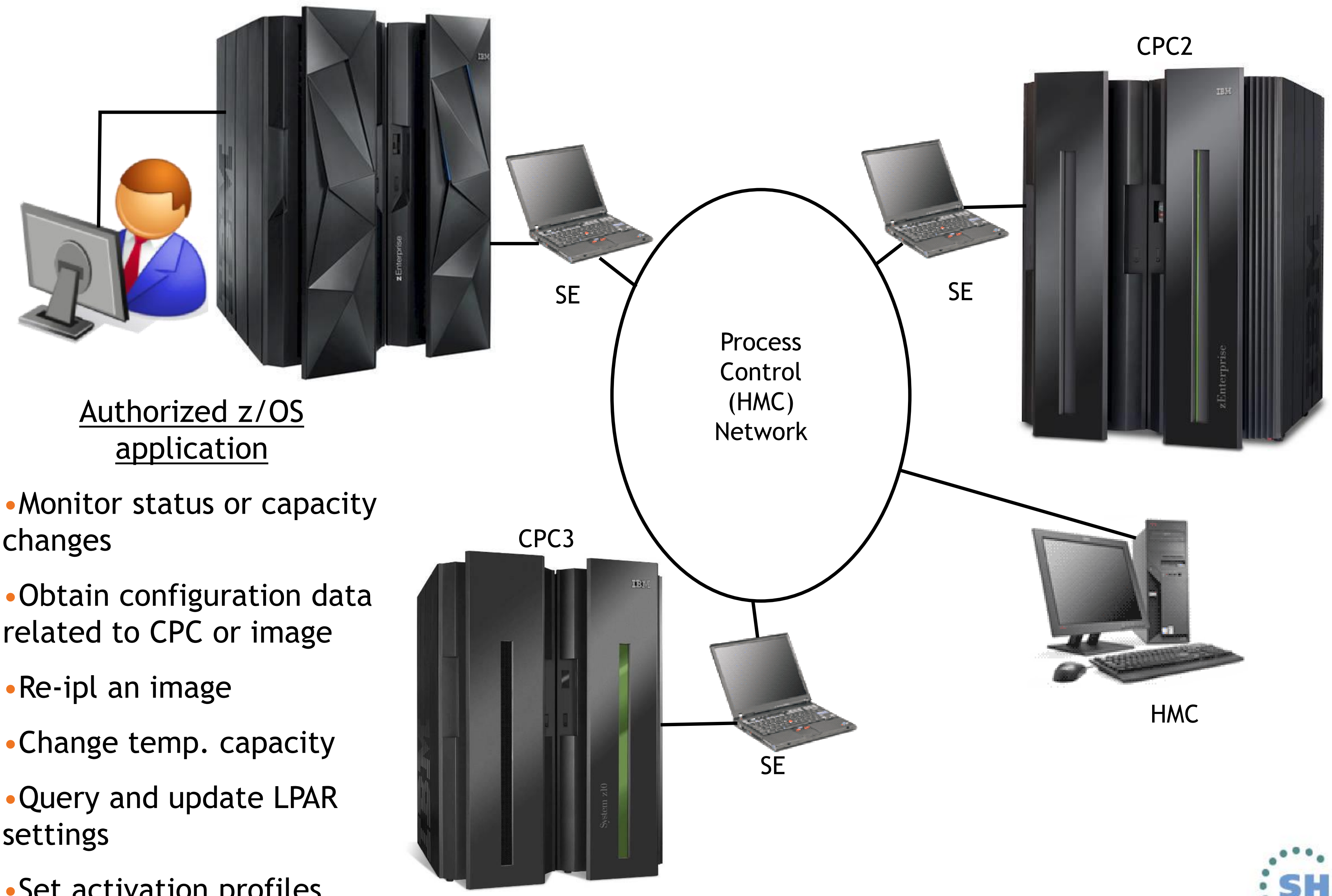
Trademarks

Agenda

- BCPii Overview
- BCPii New Features in z/OS V2R1
  - REXX support
  - Faster data retrieval
- Question and Answer Time

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

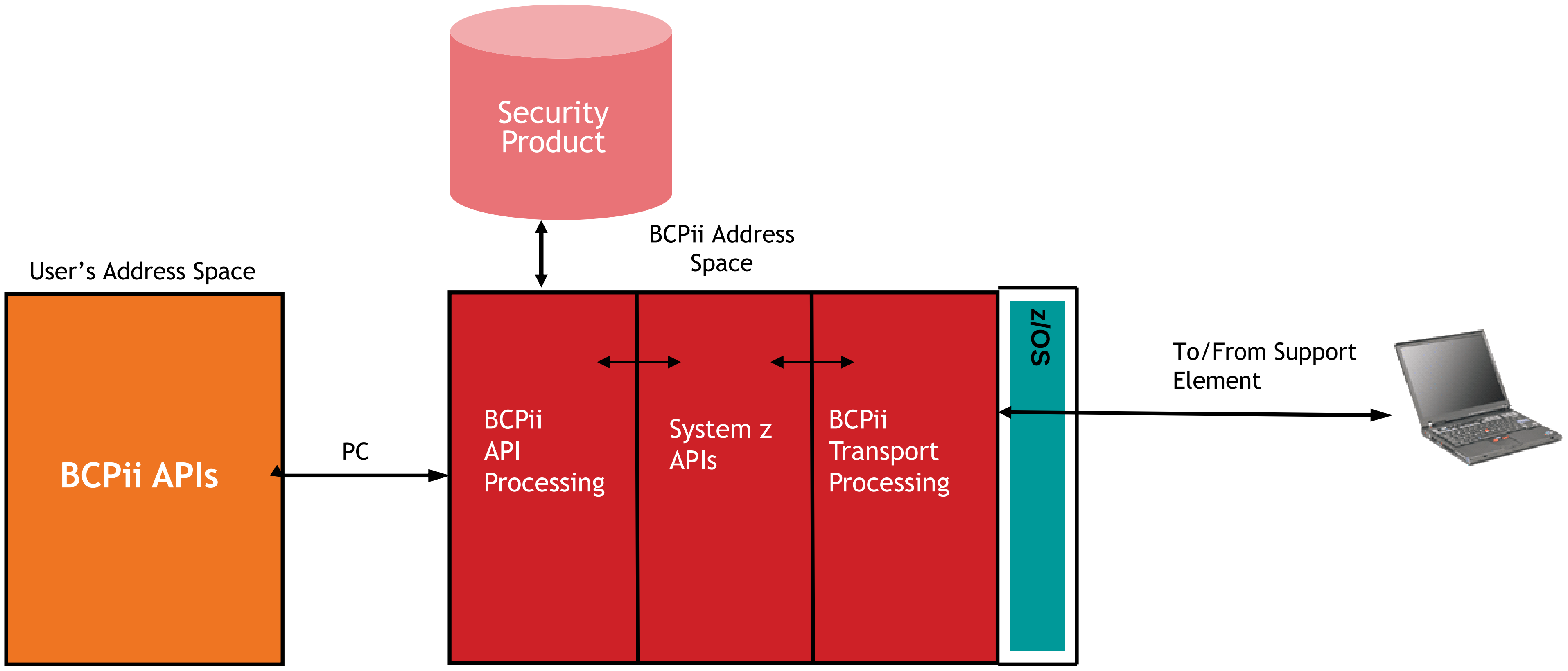
Overview - What is BCPii?

- 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

Overview - Who uses BCPii?

- z/OS operating system components
  - System Status Detection (SSD) provided by Parallel Sysplex (XCF)
  - Capacity Provisioning Manager (CPM)
  - Hardware Configuration Definition (HCD)
- Vendor applications
  - Control center, system management applications
  - Several GA'ed already
- In-house (customer-written) applications

Overview - What do the internals of BCPii look like?



BCPii Prerequisites

- **Software:**
  - Any supported z/OS release (introduced in V1R11)
- **Hardware:**
  - The program *issuing* the BCPii calls must be running on any CPC supported by z/OS
    - It is always wise to keep CPCs (even old ones) at current microcode levels
  - The target of the BCPii request can be almost any level of System z hardware

BCPii REXX Support in V2R1

z/OS BCPii Programming Environment

- 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)

The need for REXX

- 3 Marketing Field Requirements requesting REXX support in z/OS BCPii
- Numerous informal customer requests

- Why so many REXX requests?
  - "A REXX API would open the BCPii to a much wider audience by lowering the skill barrier."
  - "This would help increase adoption of the technology through less expensive development and maintenance costs when using BCPii in conjunction with existing z/OS technologies."

New BCPii REXX Support

- **BCPii is providing a new REXX host command environment for System REXX, TSO REXX and ISV REXX environments. (address bcpii)**
  - What is host command environment?
    - An environment for executing commands of a specific nature
    - Before an exec runs, an active host command environment is defined to handle commands issued by the exec.
    - When the REXX language processor encounters a command, it passes the command to the host command environment for processing
  - Example: `address bcpii “hwilist parm1 parm2 etc”`
  - BCPii’s host command environments are:
    - “Built-in” for System REXX and TSO
    - Definable for ISV-provided REXX environments

New BCPii REXX Support

- **Characteristics of the BCPii REXX host command environment:**
  - 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.
    - RC contains the REXX-specific status of request
    - ReturnCode parameter contains the BCPii status of request

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

New BCPii REXX Support

- **Characteristics of the BCPii REXX host command environment *(Continued)*:**
  - Stem variables are used extensively through BCPii REXX
    - x.0 element on lists of returned parameters indicates the number of elements in the list
    - X.0 element on lists of things specified by the user is required to be set by the user.
    - Diag Area for all services, command structures on the HWICMD, and event ids on the HWIEVENT use stem variables to make the interaction with BCPii as simple and straightforward as possible
    - The tail names of the stem variable are constants which must match the parameter names listed in HWICMD and HWIQUERY documentation.

Simple BCPii REXX Example (HWILIST)

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
```

HWICONN Example

```
ConnectType = HWI_CPC
ConnectTypeValue = CPCList.i       /* If in a loop for all CPCs  */
address bcpii "hwiconn
               ReturnCode
               InConnectToken
               OutConnectToken
               ConnectType
               ConnectTypeValue
               DiagArea."
/* If HWICONN fails, report diagnostic information.
 */
If rc <> 0 | ReturnCode <> 0 Then
  Do
    say '        Diag_Index    = '  DiagArea.Diag_Index
    say '        Diag_Key      = '  DiagArea.Diag_Key
    say '        Diag_Actual   = '  DiagArea.Diag_Actual
    say '        Diag_Expected = '  DiagArea.Diag_Expected
    say '        Diag_CommErr  = '  DiagArea.Diag_CommErr
    say '        Diag_Text     = '  DiagArea.Diag_Text
    say
  End
```

HWIQUERY Example

```
QueryParm.0 = 2
QueryParm.1.ATTRIBUTEIDENTIFIER = HWI_MMODEL
QueryParm.2.ATTRIBUTEIDENTIFIER = HWI_SNAADDR

address bcpii "hwiquery
               ReturnCode
               ConnectToken
               QueryParm.
               DiagArea."

If rc <> 0 | ReturnCode <> 0 Then
  /* Error handling code here */
Else
  Do
    say 'MModel      = ' QueryParm.1.ATTRIBUTEVALUE
    say 'SNAAddr     = ' QueryParm.2.ATTRIBUTEVALUE
  End
```

HWISET Example

```
SetType = HWI_ACCSTAT
SetTypeValue = HWMCA_STATUS_OPERATING

address bcpii "hwiset
               ReturnCode
               ConnectToken
               SetType
               SetTypeValue
               DiagArea."

If rc <> 0 | ReturnCode <> 0 Then
  /* Error handling code here */
Else
  Do
    /* Successful request processing */
  End
```

HWIEVENT Example

```
EventAction = Hwi_Event_Add
EventIDs.                          = 0
EventIDs.Hwi_Event_CmdResp         = 1
EventIDs.Hwi_Event_DisabledWait    = 1
EventExitMode = Hwi_Event_Task
EventExitAddr = ExitAddr
EventExitParm = “”

address bcpii "hwievent
               ReturnCode
               ConnectToken
               EventAction
               EventIDs.
               EventExitMode
               EventExitAddr
               EventExitParmAddr
               DiagArea."
```

HWICMD Example

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

address bcpii "hwicmd
               ReturnCode
               ConnectToken
               CmdType
               CmdParm.
               DiagArea."
```

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

BCPii System REXX support

- 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)

BCPii System REXX support

- HWIREXX invocation example

```
//STEP1    EXEC PGM=HWIREXX,REGION=1M,
//   PARM=('NAME=TESTEXEC',
//              'DSN=MY.DSN.OUTPUT',
//              'TSO=Y',
//              'SYNC=Y',
//              'TIMELIM=Y',
//              'TIME=40')
//*
//STEPLIB  DD  DSN=SYS1.LINKLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//
```

BCPii System REXX support

- AXREXX macro invocation example

```
AXREXX REQUEST=EXECUTE,TSO=NO,NAME=kExecName,                    +
       RETCODE=AxrRc,RSNCODE=AxrRsn,                             +
       TIMELIMIT=NO,                                             +
       REXXOUTDSN=kOutDsn,REXXOUTMEMNAME=kMemName2,              +
       REXXARGS=MyArgList,                                       +
       REXXDIAG=MyAxrDiag,SYNC=YES,                              +
       MF=(E,AUTOAXREXX,NOCHECK)
```

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)`

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")`
- Install APAR OA43067 when available

Programming 101 - Programming Environment (New REXX Support)

- z/OS BCPii APIs supported:

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		

Helps for writing your own BCPii REXX exec

- REXX support files *(provided in SYS1.MACLIB)*
  - HWICIREX – Main BCPii include file
  - HWIC2REX – Additional constant definitions include file
- Example:

```
HWI_LIST_CPCS                 =   1 /*                     '00000001'x  */
                                    /*    List of CPCs type             */
HWI_LIST_IMAGES               =   2 /*                     '00000002'x  */
                                    /*    List of images type           */
HWI_LIST_EVENTS               =   3 /*                     '00000003'x  */
                                    /*    List of previously subscribed*/
                                    /*    events                        */
```

Helps for writing your own BCPii REXX exec

- 2 ways of using these include files in your REXX exec:
  - If TSO=YES specified on AXREXX macro or indicated on the HWIREXX parms, then:
    - Can perform I/O and read the include files in at execution time, and interpret each line
  - If TSO=NO specified on AXREXX macro or indicated on the HWIREXX parms, then:
    - Need to copy the include file into each REXX exec or a subset of the include files of the definitions needed.

Helps for writing your own BCPii REXX exec

- BCPii REXX sample programs *(provided in samplib)*:
  - 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

Additional Configuration for BCPii REXX Support

- System REXX enablement of z/OS BCPii host command environment:
  - Verify current AXRxx parmlib member is configured correctly. (e.g.   (parameters such as REXXLIB and AXRUSER)
- TSO/E enablement of z/OS BCPii host command environment:
  - Current IKJTSOxx parmlib member must be updated with:

```
    AUTHTSF NAMES(HWIC1TRX)
```

- Potential security product configuration changes
  - Access lists for existing security profiles will need to be updated to add all new users that now wish to use z/OS BCPii in the System REXX, ISV REXX, and TSO REXX environments.

z/OS UNIX REXX considerations

- REXX execs can be executed from the z/OS UNIX shell environment. However, the execs run unauthorized.
  - z/OS UNIX REXX execs residing in the zFS and kicked off from the shell cannot directly use the BCPii host command because it is not authorized.
  - Via the TSOCMD address environment, a REXX residing in the zFS and kicked off in the shell may invoke a REXX exec residing in a z/OS data set and run successfully under TSO.
    - If TSO has been set up to run BCPii REXX execs, the exec will run successfully.

z/OS BCPii REXX Support Restrictions

- Variable names passed to BCPii limited to 40 characters long
- Literals cannot be passed to BCPii
- Don’t touch the returned ConnectToken. Just use it!
  - The ConnectToken parameter returned on the HWICONN call and passed as input on all subsequent services contains non-displayable characters.
- MODIFY AXR command cannot be used to kick off BCPii System REXX execs
  - Any attempt to run from this environment results in a return code of HWI_REXXInvalidExecutionEnv.

New Performance Enhancement In V2R1

New Performance Enhancement in V2R1

- BCPii retrieval requests can be slow, especially when multiple attributes are retrieved.
- The connection between z/OS and the SE that z/OS BCPii uses (internal proprietary interface) has both high latency and low bandwidth. Single simple query requests can average between 0.3 and 0.5 seconds on the wall clock due to various factors.
- Example (Today):
  - HWILIST ListImage: requires n+1 HwmcaGet* requests to the SE where n = number of LPARs on the CPC
  - HWIQUERY specifying 6 attributes: requires 6 HwmcaGet requests to the SE

New Performance Enhancement in V2R1

- Solution
  - Use HwmcaGetBulk* to package requests on one request to the SE and other improved algorithms in retrieving data
  - Example (Using z/OS 2.1):
    - HWILIST ListImage: requires 1 HwmcaGetBulk request to the SE (2 the first time listing child objects for a CPC)
    - HWIQUERY specifying 6 attributes: requires 1 HwmcaGetBulk request
- Benefit
  - Significant performance improvements for certain types of z/OS BCPii requests

* - Part of the System z API

New Performance Enhancement in V2R1

- Sample Scenario #1:
  - z/OS BCPii application wishes to list all image names on a CPC (60 partitions) or query multiple attributes regarding a particular LPAR (image)
  - HWILIST (HWI_LIST_IMAGES):
    - Today: 61 HwmcaGet* calls = Approx 8.5 seconds
    - New: 1 HwmcaGet* call + 1 HwmcaGetBulk* call = Approx 0.65 seconds on 1st HWILIST and 0.27 seconds on subsequent HWILIST calls.
    - **13x up to 31x improvement**

Note: Performance benefits will vary depending on the attribute being queried, number of attributes being queried simultaneously, the load of the SE, load of the z/OS image and the hardware configuration.

* - Part of the System z API

New Performance Enhancement in V2R1

- Sample Scenario #2:
  - HWIQUERY (HWI_OPERSTAT, HWI_OSNAME, HWI_OSTYPE, HWI_OSLEVEL, HWI_SYSPLEX, HWI_PARTITIONID)
    - Today: 6 HwmcaGet* calls at approx 0.4 secs each = Approx 2.43 seconds
    - New: 1 HwmcaGetBulk* call = 0.52 secs
    - **4.7x improvement**

Note: Performance benefits will vary depending on the attribute being queried, number of attributes being queried simultaneously, the load of the SE, load of the z/OS image and the hardware configuration.

* - Part of the System z API

New Performance Enhancement in V2R1

- What do I need to do to take advantage of this?
  - No changes to z/OS BCPii configuration or applications required.
    - Must target z9 (running at latest microcode level) or higher to take advantage of performance improvements
  - Note: If HWIQUERY requests were called separately for each attribute in the past, a modification to the application to combine the attribute queries into a single HWIQUERY call can improve performance significantly

New Performance Enhancement in V2R1

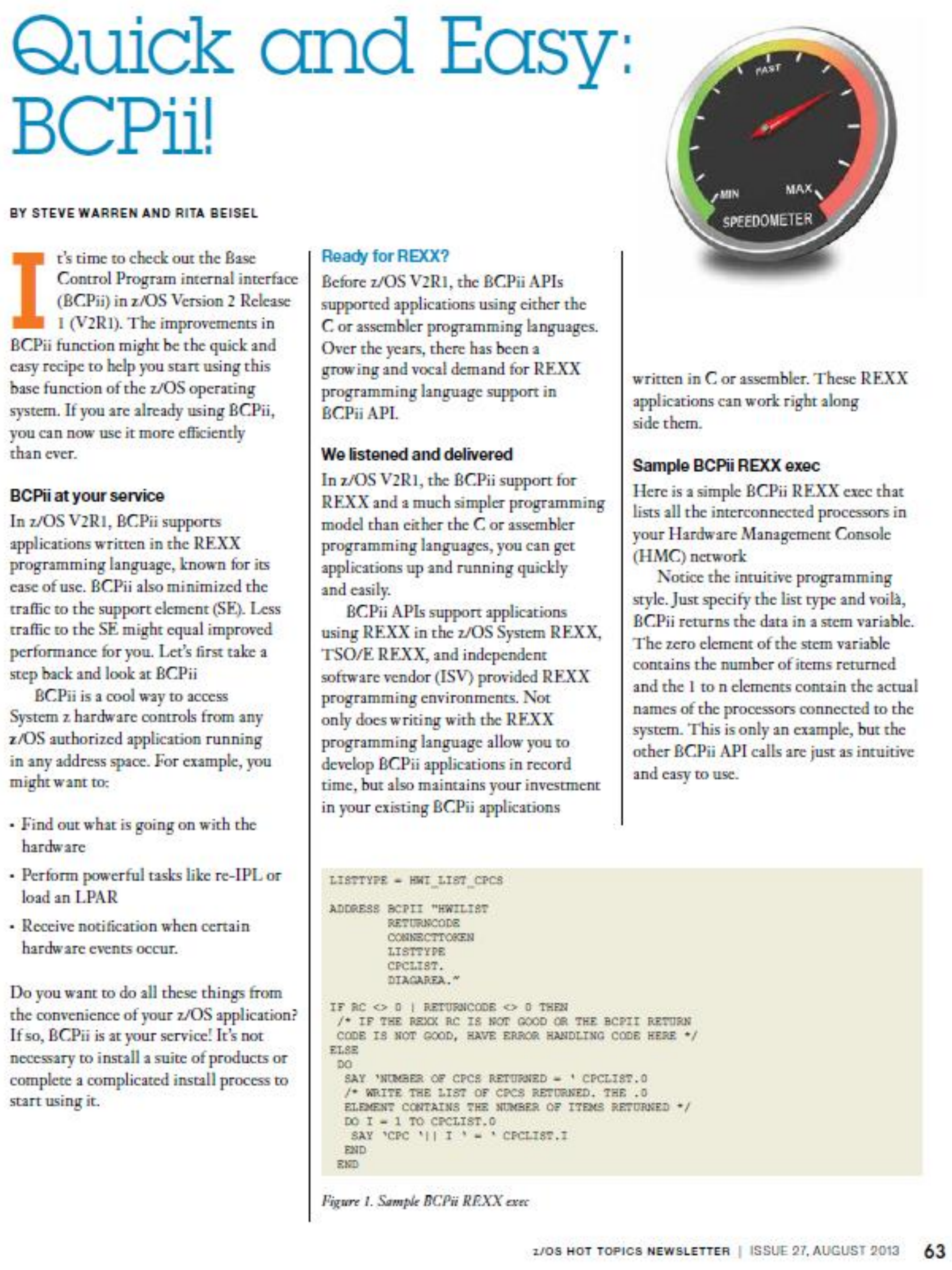
- Example:
  - Good programming technique
    - Call HWIQUERY (HWI_OPERSTAT, HWI_OSNAME, HWI_OSTYPE, HWI_OSLEVEL, HWI_SYSPLEX, HWI_PARTITIONID)
  - Not as optimal programming technique
    - Call HWIQUERY (HWI_OPERSTAT)
    - Call HWIQUERY (HWI_OSNAME)
    - Call HWIQUERY (HWI_OSTYPE)

BCPii further information

- 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 presentations regarding BCPii:
  - 13847: Recent z/OS Enhancements You Can Use to Reduce Down Time, presented by Frank Kyne and Karan Singh.
    Thursday, August 15, 2013: 1:30 PM-2:30 PM
- 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
```

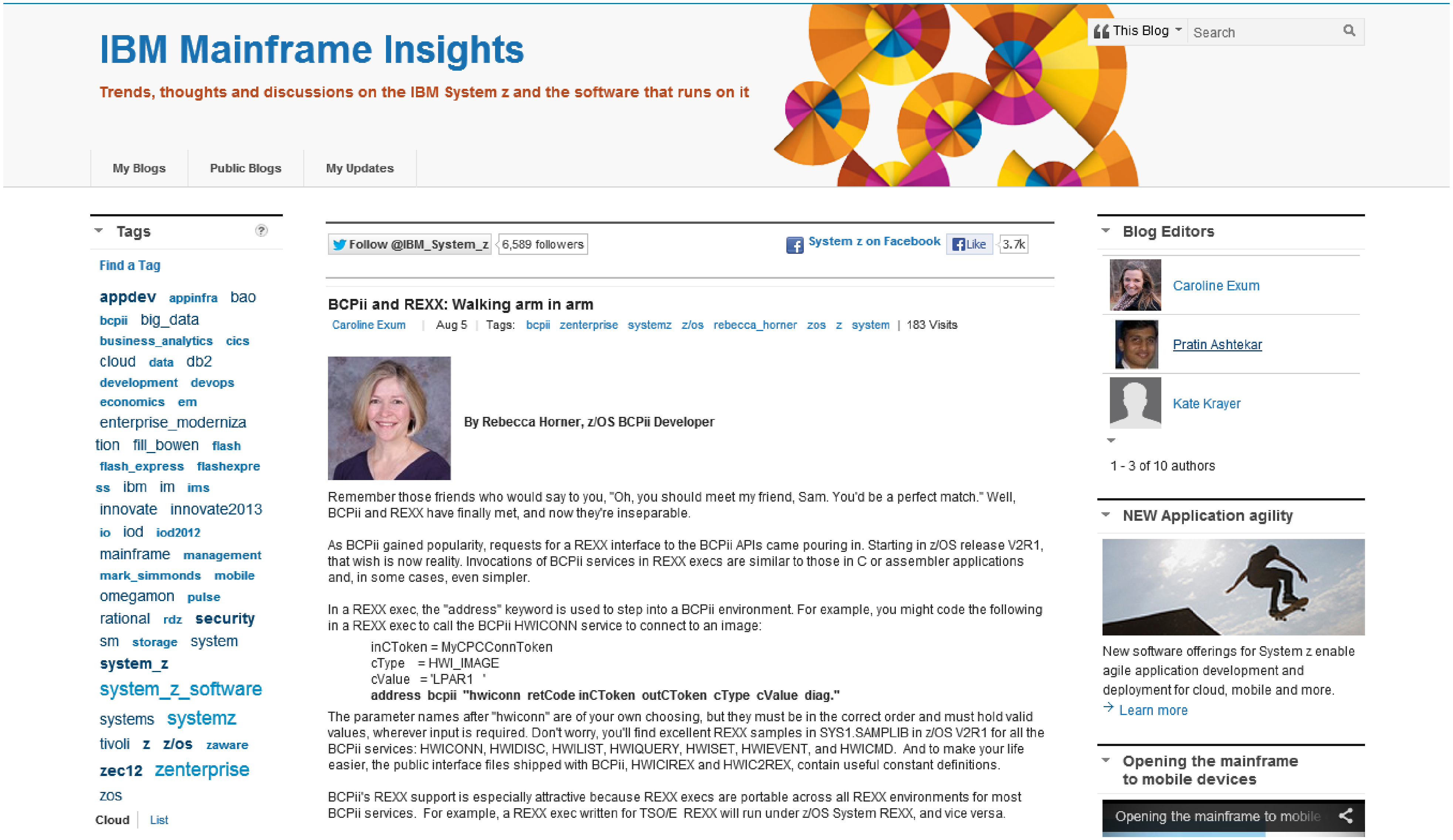
Figure 1. Sample BCPii REXX exec

New 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**

  - Some blog entries:
    - BCPii and REXX: Walking arm in arm
    - We heart z/OS BCPii
    - How about a slice of BCPii? (Discussion of BCPii samples)
    - News Flash: z/OS BCPii Summary Tables Wanted!
    - 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

This Blog
Search
My Blogs
Public Blogs
My Updates
Tags
Find a Tag
appdev appinfra bao bcpii big_data business_analytics cics cloud data db2 development devops economics em enterprise_moderniza tion fill_bowen flash flash_express flashexpre ss 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.

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

Questions?