IBM WebSphere Developer for zSeries Version 6.0.1

WebSphere Developer for zSeries Host Configuration Guide

SC31-6930-00

IBM WebSphere Developer for zSeries Version 6.0.1

WebSphere Developer for zSeries Host Configuration Guide

SC31-6930-00

Note

Before using this document, read the general information under "Notices" on page 41.

First edition (October 2005)

This edition applies to IBM WebSphere Developer for zSeries (product number 5724-L44) and to all subsequent releases and modifications until otherwise indicated in new editions.

Order publications by phone or fax. IBM Software Manufacturing Solutions takes publication orders between 8:30 a.m. and 7:00 p.m. eastern standard time (EST). The phone number is (800) 879-2755. The fax number is (800) 445-9269. Faxes should be sent Attn: Publications, 3rd floor.

You can also order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

IBM welcomes your comments. You can send your comments by mail to the following address:

IBM Corporation, Attn: Information Development, Department 53NA Building 501, P.O. Box 12195, Research Triangle Park, NC 27709-2195.

Contents

About this book

This book discusses the Configuration of the WebSphere® Developer for zSeries® functions. It includes instructions on how to configure WebSphere Developer servers on your z/OS® host system.

Note: The configuration information found in this document is for WebSphere Developer for zSeries Version 6.0.1. For previous versions of WebSphere Developer and WebSphere Studio Enterprise Developer, use the configuration information found in the Program Directories for those releases.

Who should read this book

This book is intended for system administrators who would like to install and configure WebSphere Developer for zSeries Version 6.0.1 on their z/OS host system. To use this book, you need to be familiar with the USS and MVS™ host systems.

Chapter 1. Installing the host components

For each of the following functions, install the required FMIDs. For installation information about the various FMIDs, please refer to the corresponding program directory for the FMID you are installing.

Table 1. WebSphere Developer installation and configuration matrix

If you require this WebSphere Developer for zSeries function	You must install these FMIDs	And you will find configuration information here
• Host Connectivity • JES Connectivity	H001600, H002600	• Chapter 2, "Activating IBM WebSphere Developer for zSeries RSE+ICU," on page 3 • Chapter 3, "Activating IBM WebSphere Developer for zSeries JES Job Monitor," on page 11.
• RSE Remote Compile • Error Feedback for Remote Compile • Remote Debugging • DB2® Stored Procedures • BMS Map Support • SCLM Access	HEDS500, H001600, H002600	• Chapter 4, "Activating IBM WebSphere Studio Enterprise Developer Options for z/OS," on page 13 • Chapter 2, "Activating IBM WebSphere Developer for zSeries RSE+ICU," on page 3 • Chapter 3, "Activating IBM WebSphere Developer for zSeries JES Job Monitor," on page 11
• Common Software Configuration Management Access	H001600, HCMA601	• Chapter 2, "Activating IBM WebSphere Developer for zSeries RSE+ICU," on page 3 • Chapter 5, "Activating the IBM WebSphere Developer for zSeries Common Access Repository Manager," on page 35
• Enterprise Generation Language (EGL)	HEDS500, H284500	• Chapter 4, "Activating IBM WebSphere Studio Enterprise Developer Options for z/OS," on page 13 • The EGL documentation • The Program Directory for H284500.
• Bidirectional language support	HBDI601	• Chapter 6, "WebSphere Developer for zSeries Bidirectional support," on page 39

Chapter 2. Activating IBM WebSphere Developer for zSeries RSE+ICU

Refer to *Program Directory for IBM® WebSphere Developer for zSeries RSE + ICU* (GI10-3358, FMID:H001600) for the SMP/E installation instructions for RSE + ICU.

Before installing the 6.0.1 service, if you were a 6.0 user, it is recommended that you save the 6.0 customization. For example, The three files that need to be backed up for H001600 are:

- rsed.envvars
- setup.env.zseries
- server.zseries

To get to Version 6.0.1 you must install the PTF for FMID H001600 (the PTF on the WebSphere Developer for zSeries Version 6.0.1 host CD).

Customizing the IBM WebSphere Developer for zSeries RSE+ICU components installed in the USS

Pre-configuration considerations

1. If you are a WebSphere Developer Version 6.0 user, it is recommended that you save the 6.0 customization files referenced above before installing the 6.0.1 service.
2. Java 1.4.2 is required and must have read access in a USS HFS directory.
3. APPC must be active.
4. INETD must be active.

   Note: If INETD is started by JCL using BPXBATCH, region size must be 0. If INETD is started from a TSO/OMVS session, the TSO region size must be 2096128.
5. The TSO Userids that will use the WebSphere Developer workbench to access MVS, USS and/or JES must have the following:
   a. An OMVS segment defined to the security system, for example, RACF, unless a default OMVS segment is used. See ″z/OS(R) UNIX(R) System Services Planning GA22-7800″ for the default OMVS segments. See the following example.
   b. A home directory allocated for the user and identified in the OMVS segment. See the following example.
   c. In the userid's OMVS segment the PROGRAM field should be '/bin/sh' or other valid USS shell such as '/bin/tcsh'. See the following example.
   d. Userids require UID 0 if you manually start INETD from USS. Userids.do not require UID 0 to access MVS from the workbench

      Example:

```
OMVS INFORMATION
----------------
UID= 0000003200
HOME= /u/userid
```

```
PROGRAM= /bin/sh
CPUTIMEMAX= NONE
ASSIZEMAX= NONE
FILEPROCMAX= NONE
PROCUSERMAX= NONE
THREADSMAX= NONE
MMAPAREAMAX= NONE
```

6. Set MAXASSIZE in BPXPRMxx parmlib member to 2147483647.

   This can be checked and set dynamically(until the next IPL) with the following commands (as described in z/OS MVS System Commands SA22-7627):

   - DISPLAY OMVS,O

     SETOMVS MAXASSIZE=2147483647

7. CBC.SCLBDLL (the C++load library) Must be in LINKLIST and APF-authorized.
8. Depending on the local installation standards for TCP/IP, a /etc/hosts file will likely be needed.

   Example

   ```
   #  Resolver /etc/hosts file ctfmvs08
   9.42.111.75   ctfmvs08 # CTFMVS08 Host
   9.42.111.75   ctfmvs08.rtp.raleigh.ibm.com # CTFMVS08 Host
   127.0.0.1     localhost
   ```

9. JES Job Monitor Program Number 5724-L44 FMID H002600 and service, must be installed and activated for certain functions to be available to RSE WebSphere Developer workbench clients. These functions are:
   - JES Job Submission
   - JES Job Monitoring
   - JCL Submission

Program Control Authorization:

In /usr/lpp/wd4z/rse/lib (RSE installation directory) mark these files as Program Controlled by issuing the following commands in OMVS:

- `extattr +p fekfdir.dll`
- `extattr +p fekfivp`
- `extattr +p fekflock`
- `extattr +p fekfrsed`

Check by issuing `ls -laE fek*` to display the attributes.

- `fekfscmd` does not require the `+p`

Example output:

```
# ls -laE fek*
-rwxr-xr-x -ps- 2 DFS SYS1 188416 Apr 26 9:35 fekfdir.dll
-rwxr-xr-x -ps- 1 DFS SYS1 114688 Jun 1 16:43 fekfivp
-rwxr-xr-x -ps- 2 DFS SYS1 13172 Apr 26 9:34 fekflock
-rwxr-xr-x -ps- 2 DFS SYS1 147456 Apr 26 9:34 fekfrsed
-rwxr-xr-x --s- 2 DFS SYS1 42521 Jul 26 11:59 fekfscmd
```

RSE Daemon and REXEC set up

You may run RSE from an RSE Daemon and/or REXEC USS server port. You need to customize either or both ways depending on how your users plan to operate.

Both the REXEC method and the RSE Daemon method use the variables set in the rsed.envvars file in the installation directory, rsed.envvars must be customized for either method.

Note: INETD and/or the REXEC UNIX® Command Server must have been previously enabled.

You may find the following publications helpful:

- *IBM z/OS Communications Server IP Configuration Guide SC31-8775*
- *IBM z/OS Communications Server IP Configuration Reference SC31-8776*
- *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 2: UNIX Applications* Website
  http://www.redbooks.ibm.com/redbooks/SG245228.html

INETD RSE Daemon set up (to run from an INETD RSE daemon):

rsed.envvars Customization

In /usr/lpp/wd4z/rse/lib/rsed.envvars make these modifications

1. Modify `JAVA_HOME=/usr/lpp/java/J1.4` to the correct location for the JAVA directory
2. Modify `ICU_HOME=/usr/lpp/wd4z/icuc` to the directory where ICU is installed.
3. Do not modify the `CARMA_HOME=.` line, it's use is documented in Common Access Repository Manager Program Number 5724-L44 FMID HCMA601.
4. Modify `TZ=EST5EDT` to your MVS system's time zone.
5. Modify `PATH=/bin:/usr/sbin:$JAVA_HOME/bin:$RSE_LIB:`, if needed.

Adding RSE Daemon to INETD

1. Verify INETD is running, this can be done with a ps -e command
2. The output from ps -e should contain a reference to inetd, e.g. /usr/sbin/inetd
3. Modify /etc/services by adding this line:

   ```
   rse       4035/tcp       RADz RSE
   ```

   Note: 4035 is the default port in the WebSphere Client; Properties for MVS Files; Server Launcher Settings; Remote Daemon Port.
4. Modify /etc/inetd.conf by adding the following line:

   ```
   "rse stream tcp nowait OMVSKERN /usr/lpp/wd4z/rse/lib/fekfrsed
                                          rsed -d /usr/lpp/wd4z/rse/lib"
   ```

   Note: Quotes used in the previous example to signify one continuous line.
5. To activate the changes made to the /etc files you must take some action, for example an IPL or stopping/starting INETD will activate the changes.

Installation verification of the RSE daemon.

1. From a USS command prompt while in the `/usr/lpp/wd4z/rse/lib` directory issue the following command:

   ```
   ./fekfivp 4035 userid
   ```

   Where userid is a valid userid. It will prompt you for the password. Sample output:

```
./fekfivp 4035 userid
Password:
connected
8108


Success
```

2. The daemon port availability can be verified by issuing ″netstat″ from a USS prompt. Result should include a listing similar to:

```
INETD4     00000030 0.0.0.0..4035          0.0.0.0..0           Listen
```

INETD REXEC (USS) set up (to run using REXEC)

Verify INETD REXEC is Running

A common port used by REXEC is 512. This can be verified with a netstat command from a USS command line. In the netstat output there should be a line like:

```
INETD4     0000002E  0.0.0.0..512     0.0.0.0..0        Listen
```

Note: Both the REXEC method and the RSE Daemon method use the variables set in the rsed.envvars. To run from the INETD USS REXEC port, the rsed.envvars file must be updated following the instructions in “rsed.envvars Customization”

Check the Installation: Check the installation by executing the following command in `/usr/lpp/wd4z/rse/lib/`:

```
/bin/sh server.zseries
```

or

```
./server.zseries
```

The command was successful if it returns a port number.

Sample output:

```
# Server Started Successfully

3271

Server running on: MVSNAME
#
```

Defining the PORTRANGE Available for RSE (OPTIONAL)

This is a part of rsed.envvars customization that specifies the ports on the client on which the server tries to connect.

- To specify the port range, for the client to communicate with z/OS, change the following line in rsed.envvars:
  - `_RSE_PORTRANGE=8108-8118`

  The format of PORTRANGE is : `_RSE_PORTRANGE=min-max (e.g. 8108-8118)`
- The port number used by the RSE server is determined in the following order:

1. If a non-zero port number is specified in the subsystem properties, the port number is used. If the port is not available connect will fail
2. If a port number in the subsystem properties is 0, and if _RSE_PORTRANGE is specified in rsed.envvars, the port range specified by _RSE_PORTRANGE is used. If no port in the range is available, connect will fail.
3. If a port number in the subsystem properties is 0, and _RSE_PORTRANGE is not specified in rsed.envvars, any available port is used.

Notes:

1. The port range specifies minimum port number and (maximum port number + 1). e.g. `_RSE_PORTRANGE=8108-8118` means port numbers from `8108` up to `8117`
2. When a server opens a port and is listening, the port number cannot be used by another server, but once it is connected, the same port number can be used. This means that the number of ports in the range does not limit the number of users connected concurrently.

Defining the APPC Transaction for the TSO Commands Service

The TSO Commands Service is implemented as an APPC transaction program, FEKFRSRV, and must be active for RSE connections to succeed between the host and the client. FEKFRSRV acts as a host server to execute TSO commands that are issued from the workstation through TCP/IP.. APPC is not required on the workstation because the workstation communicates with FEKFRSRV through TCP/IP.

Each workstation can have an active connection to multiple hosts at the same time.

The following steps are in preparation for configuring the TSO Commands Service on the host.

You (the system programmer or APPC administrator) may have completed the first step.

1. Install, configure, and start TCP/IP on your z/OS system.
2. Verify access through TCP/IP by using the TSO command HOMETEST. TCP/IP must be able to respond to a GETHOSTBYNAME request. Therefore, a domain name server must exist and know the name of the MVS TCP/IP stack that has access to it, or the domain name must be defined to TCP/IP directory.
3. The system programmer or APPC administrator needs to do the following steps to configure the command facility:
   a. Start APPC and the APPC transaction scheduler (ASCH) subsystem, which handles incoming requests for local transaction programs.
   b. Define the APPC transaction that will act as a command server. You can use the sample JCL hlq.SFEKSAMP(FEKFRTAD) provided to define this transaction.

      Sample JCL is also provided to display hlq.SFEKSAMP(FEKAPPCL) or delete hlq.SFEKSAMP(FEKAPPCX) the transaction
   c. Define the scheduling information for the APPC transaction scheduler by using the ASCHPMxx member of SYS1.PARMLIB. Include a definition of the class to be used by the transaction program FEKFRSRV. The class for FEKFRSRV is set in FEKFRTAD. Therefore the class in FEKFRTAD must match the class defined in SYS1.PARMLIB(ASCHPMxx).

```
For example:
CLASSADD
CLASSNAME(A)
MAX(2)
MIN(1)
MSGLIMIT(5)
OPTIONS
DEFAULT(DEFAULT)
TPDEFAULT
REGION(2M)
TIME(5)
MSGLEVEL(1,1)
OUTCLASS(9)
```

d. Control the dispatching priority of the FEKFRSRV transaction program by using parameters for the systems resource manager (SRM) to associate FEKFRSRV with a domain and performance group. Because FEKFRSRV issues TSO commands, it should be assigned to a TSO performance group.

e. Define a default OMVS segment for the system or for each user who needs to use remote edit-compile-debug.

f. Make the following parts available on the host system:
   - FEKFRTAD (the JCL that defines the APPC server transaction) in SFEKSAMP
   - FEKFSERV (REXX CLIST for the command server), in SFEKSAMP

g. If you cannot use the command facility, there are two areas where problems may arise: connecting to MVS and starting the APPC server transaction.
   - If you do not see the messages about setting up APPC, check the system log for RACF® messages or other messages related to the command that was issued or the user ID that issued it.

     Common causes of problems include:
     - You do not have read authority to the SFEKSAMP data set.
     - TCP/IP is not active, or MVS is unreachable (not pingable) due to network problems, a bad IP address, or other problems.
   - If you see the messages about setting up APPC but do not see the message confirming that setup succeeded, the APPC server transaction was probably unable to start. If the error log (userid.FEKFRSRV.&TPDATE.;&TPTIME.LOG;) for the conversation exists, check it. Some of the likely causes of problems are:
     - The TCP/IP stack is not using the default name of TCPIP and the SYSTCPD DD card has not been set or is pointing to the wrong data set.
     - The server was unable to allocate SYSPROC or SYSTSPRT.
     - The JCL points to the wrong SYSPROC, (SYSPROC needs to point to the SFEKSAMP data set.)
     - The server could not open or access the message_data_set.
     - There are not enough APPC scheduler initiators available.
     - APPC or ASCH address spaces are not active.
     - The class named "A" is not defined to the APPC scheduler ASCH.
     - There is no default OMVS segment for the system, and the user does not have an OMVS segment.

Verification of the TSO Commands APPC transaction:

From the RSE lib directory (/usr/lpp/wd4z/rse/lib) run the following:

- ===> ./fekfscmd '1.2.3.4 * notrace USERID PASSWORD'

Note: In the above command replace USERID & PASSWORD with valid userid & password values.

Should present a listing similar to the following (time stamps will vary):

```
# ./fekfscmd '1.2.3.4 * notrace USERID PASSWORD'
20050420 17:17:47.815791 ./fekfscmd: version=Oct 2003
20050420 17:17:47.816022 Input parms: 1.2.3.4 * notrace USERID *****
20050420 17:18:16.099886 APPC: Allocate succeeded
20050420 17:18:16.100062 Conversation id is 0D8BE3F80000010E
20050420 17:18:16.100323 APPC: Set Send Type succeeded
20050420 17:18:16.227847 APPC: Confirm succeeded
20050420 17:18:16.228103 Req to send recd value is 0
20050420 17:18:16.228454 APPC: SEND_DATA return_code = 0
20050420 17:18:16.228603 request_to_send_received = 0
20050420 17:18:16.228744 Send Data succeeded
20050420 17:18:16.228974 APPC: Set Prepare to Receive type succeeded
20050420 17:18:16.229289 APPC: Prepare to Receive succeeded
20050420 17:18:16.444467 APPC: Receive data
20050420 17:18:16.444729 RCV return_code = 0
20050420 17:18:16.444866 RCV data_received= 2
20050420 17:18:16.444995 RCV received_length= 29
20050420 17:18:16.445118 RCV status_received= 4
20050420 17:18:16.445246 RCV req_to_send= 0
20050420 17:18:16.445375 Receive succeeded
:IP: 0 9.42.111.75 2527 62296
20050420 17:18:16.446294 APPC: CONFIRMED succeeded
```

For information on diagnosing RSE daemon connection problems, see the technotes on the WebSphere Developer for zSeries Support Page (*http://www-306.ibm.com/software/awdtools/devzseries/support/*). To get to the information, select **Self help** -> **Solve a problem** -> **technotes**.

Chapter 3. Activating IBM WebSphere Developer for zSeries JES Job Monitor

Refer to *Program Directory for IBM WebSphere Developer for zSeries JES Job Monitor* (GI10-3359, FMID:H002600) for the SMP/E installation instructions for JES Job Monitor.

Customizing the IBM WebSphere Developer for zSeries JES Job Monitor Configuration file

If you are a 6.0 user, it is recommended that you save the 6.0 customization contained in the configuration file FEJJCNFG before installing the 6.0.1 service. The configuration file is FEJJCNFG is located in hlq.SFEJSAMP.

To get to Version 6.0.1 you must install the PTF for FMID H002600. The PTF came on the 6.0.1 host CD.

Follow these steps to complete the activation/configuration of IBM WebSphere Developer for zSeries JES Job Monitor. The configuration file is FEJJCNFG and is located in hlq.SFEJSAMP. The file contains:

```
SERV_PORT=6715
CODEPAGE=UTF-8
HOST_CODEPAGE=IBM-1047
JESNAME=JES2
TZ=EST5EDT
#LIMIT_VIEW=USERID
LISTEN_QUEUE_LENGTH=5
MAX_DATASETS=32
MAX_THREADS=200
TIMEOUT_INTERVAL=1200
TIMEOUT=3600
AUTHMETHOD=RACF
```

1. SERV_PORT=; The default port is (6715). Change as desired, however BOTH the client and the server must be configured with the same port number. If you change the server port number, all WebSphere Developer for zSeries client users must also change the JES Job Monitor Port for systems in Remote Systems View.
2. CODEPAGE=; The default is UTF-8. The workstation codepage is set to UTF-8 and generally should not be changed. You might need to change UTF-8 to match the workstation's codepage if you have difficulty with NLS characters, such as the currency symbol.
3. HOST_CODEPAGE=; The default is (IBM-1047). Change as needed.
4. JESNAME=; The default is JES2. Use the JESNAME parameter in the configuration file to indicate whether you use JES2 or JES3.
5. TZ=; The default is EST5EDT. The default time zone is UTC +5 hours (Eastern Standard Time (EST) Eastern Daylight Savings Time (EDT)). Change this to represent your time zone. Additional information may be found in the z/OS UNIX System Services File System Interface Reference SA22-7808.
6. LIMIT_VIEW=; The default allows viewing of all output. Defines what output the user can view. Specifying USERID limits the view to output owned by the

user. If USERID is not specified the user will be able to view everyone's output. Note: The only valid setting is USERID.

7. LISTEN_QUEUE_LENGTH=; The TCP/IP listen queue length. The default is 5. Do not be changed unless directed to do so by IBM service.
8. MAX_DATASETS=; The default is 32. This is the maximum number of spooled output datasets that JES Job Monitor will return to the client. (i.e. SYSOUT, SYSPRINT, SYS00001, etc.).
9. MAX_THREADS=; The default is 200. Maximum number of users that can be using one JES Job Monitor at a time. Increasing this number may require you to increase the size of the WebSphere Developer for zSeries JES Job Monitor address space.
10. TIMEOUT_INTERVAL=; The default is 1200. Controls how often the server checks for and kills threads which have timed out (see TIMEOUT).
11. TIMEOUT=; The default is 3600. The length of time before a thread is killed due to lack of interaction with the client.
12. AUTHMETHOD=; The default is RACF. Means that the standard security interface is used. Should not be changed, even if you use a product other than RACF.

CONCHAR may optionally be added to specify the JES console command character. CONCHAR defaults to CONCHAR=$ for JES2, or CONCHAR=* for JES3. If your installation has defined a different command character, specify it as the value for CONCHAR.

Customizing the IBM WebSphere Developer for zSeries JES Job Monitor Startup JCL

1. Follow the customization steps in the JCL located in hlq.SFEJSAMP(FEJJJCL).
2. APF authorize the product load library, hlq.SFEJLOAD.
3. Submit the JCL to start the JES Job Monitor. You can use the netstat command to check whether the JES Job Monitor is listening to the configured port.

In order to allow users to execute operations via the JES Job Monitor, they must be given access authority to the OPERCMDS class. This can be done conditionally, so users' access is only in effect when they are using the JES Job Monitor. To use this conditional access checking, you must have the CONSOLE class active and the JMON console defined in the CONSOLE class.

For example, you would issue the following RACF commands:

```
SETROPTS CLASSACT(CONSOLE)
RDEFINE CONSOLE JMON UACC(READ)
```

Use the following RACF commands to permit users to issue JES2 commands only while running under the JES Job Monitor:

```
RDEFINE OPERCMDS JES2.** UACC(NONE)
PERMIT CLASS(OPERCMDS) JES2.** ID(userid or groupid) ACCESS(CONTROL)
WHEN(CONSOLE(JMON))
```

Chapter 4. Activating IBM WebSphere Studio Enterprise Developer Options for z/OS

Refer to *Program Directory for IBM WebSphere Developer for zSeries Enterprise Developer Options for z/OS* (GI10-3242, FMID: HEDS500) for the SMP/E installation instructions for Enterprise Developer Options for z/OS.

Note: This configuration information is for WebSphere Developer for zSeries Version 6.0.1. For previous versions of WebSphere Developer and WebSphere Studio Enterprise Developer, use the configuration information found in the Program Directories for those releases.

Before installing the WebSphere Developer for zSeries Version 6.0.1 service, if you were a WebSphere Developer Version 6.0 user, it is recommended that you save the 6.0 customization. To get to Version 6.0.1 you must install the PTFs for FMID HEDS500. The PTFs come on the 6.0.1 host CD.

After the product has been installed on your z/OS system, some customization is required before additional verification procedures can be run and Enterprise Developer Options for z/OS can be used.

Customization of Enterprise Developer Options for z/OS means tailoring the product for use in site-specific environments. This might include adding support for another product or changing default values that are currently in effect.

This section contains the information you need to customize your installation of Enterprise Developer Options for z/OS. The need to perform the tasks will vary depending on the function you are using within WebSphere Studio Enterprise Developer.

- Table 1 on page 2 lists the customization tasks in the MVS or USS environment needed to support the various functions within Enterprise Developer.
- The rest of the section consists of detailed descriptions of the customization procedures.

Customizing IBM WebSphere Studio Enterprise Developer Options for z/OS

The definitions for the support are as follows:

- z/OS Application Development Support - The tasks used to support the remote edit, compile, and debug of COBOL, Assembler and PL/I programs.
- EGL Generated COBOL support - The tasks used to support the build of COBOL programs generated using the Enterprise Generation Language.
- EGL Generated Java™ Support - The tasks used to support the build and/or execution of Java programs generated using the Enterprise Generation Language.

Notes:

1. The execution libraries for EGL generated COBOL programs are provided in the IBM Enterprise Developer Server for z/OS for z/OS V5.0.0 product

(Program No: 5655-I57/FMID: H284500.). This product must be ordered and installed to build and execute EGL generated COBOL programs. This is not included in the customization tasks.

2. The IBM Debug Tool for z/OS and OS/390® V3.1 (Program No. 5655-H32) will need to be installed and configured to support remote debug of assembler, COBOL and PL/I programs from the Enterprise Developer z/OS Application Development Tools. This can be ordered as a separate product or is included in the full function offerings of Enterprise COBOL for z/OS V3.2 or Enterprise PL/I for z/OS V3.2. This task is not included in the customization tasks.

Making the Enterprise Developer Options for z/OS Load Library APF authorized

It is possible to configure the remote build server in a way that does not require the target load library to be APF authorized. However, this is not very practical when using the build server in conjunction with Enterprise Developer for the following reasons:

1. The library must be APF authorized if the build server is to operate under the authority of the Enterprise Developer user requesting a build function rather than of the person starting the build server.
2. A non-APF authorized program cannot invoke an APF authorized program. Users of Enterprise Developer will almost surely require the build server to invoke programs in APF authorized libraries. For example, the terminal monitor program used to perform DB2 binds is usually APF authorized. The SCLM functions in the z/OS IDE must reside in APF authorized libraries.

Therefore, the target load library hlq.CCU.V5R0M0.SCCULOAD (default name) should be made APF authorized.

ELAX procedures

Enterprise Developer Options for z/OS provides sample JCL procedures which will be used for the JCL generation, remote project builds and remote syntax check features of COBOL, PLI and Assembler programs and BMS maps using WebSphere Studio Enterprise Developer.

SMP/E installs these sample JCL procedures into hlq.SCCUSAMP. To use these procedures you must copy all procedures named ELAXF* into a system proclib that is available to users. The procedures must be customized to reflect naming conventions used on the target system. The required customization is documented within each JCL procedure. These procedures allow installations to apply their standards. This will also ensure that developers use the same procedures with the same compiler options and compiler levels.

The JCL Generation/Remote project builds and remote Syntax check operations done from WebSphere Studio Enterprise Developer client assume that these procedures are customized and available to the user.

The names of the procedures and the names of the steps in the procedures match the default properties that are shipped with the client. If you decide to change the name of the procedure or the name of a step in a procedure, the corresponding properties file on the client should also be updated. We recommend that you do not change the names of the procedures and step names.

The sample procedures to be copied/customized are:

ELAXFASM
Sample procedure for assembling High Level assembler programs.

ELAXFBMS
Sample procedure for creating CICS BMS object and corresponding copy, dsect, or include member.

ELAXFCOC
Sample procedure for doing COBOL Compiles, Integrated CICS translate and integrated DB2 translate.

ELAXFCOP
Sample procedure for doing DB2 pre-process of EXEC SQL statements embedded in COBOL programs.

ELAXFCOT
Sample procedure for doing CICS translation for EXEC CICS statements embedded in COBOL programs.

ELAXFGO
Sample procedure for the GO step.

ELAXFLNK
Sample procedure for linking COBOL/PLI/High Level Assembler programs.

ELAXFPLC
Sample procedure for doing PLI Compiles, Integrated CICS translate and integrated DB2 translate.

ELAXFPLT
Sample procedure for doing CICS translation of EXEC CICS statements embedded in PLI programs.

ELAXFPLT
Sample procedure for doing CICS translation of EXEC CICS statements embedded in PLI programs.

ELAXFPLP
Sample procedure for doing DB2 pre-process of EXEC SQL statements embedded in PLI programs.

ELAXFUOP
Sample procedure for generating the UOPT step when building programs that run in CICS or IMS subsystems.

New installation customization instructions:

An ISPF configuration table ISPF.conf must be customized for ISPF dataset allocations. Copy and rename the supplied sample configuration file ELAXWEBI to /etc/EnterpriseDeveloper/SCLM/ISPF.conf, if you use a different path name you will need to update CGI_ELAXPATH webserver environment variable . The sample HTTP webserver environment variable file will have by default CGI_ELAXPATH = /etc/EnterpriseDeveloper/SCLM

Note: If the environment variable CGI_ELAXPATH is not included the program will look for the ISPF.conf file in the WSED/SCLM install path.

DB2 stored procedures

The following customization is needed to incorporate the DB2 Stored Procedures into your system.

After the SMP/E Apply of PTF UQ84053, the sample library hlq.SCCUSAMP contains the following DB2 Stored Procedure members:

- ELAXMREX REXX code of the PL/I and COBOL Stored Procedure Builder.
- ELAXMJCL Sample JCL for defining the PL/I and COBOL Stored Procedure Builder to DB2
- ELAXMSAM Sample JCL procedure of the WLM address space for the PL/I and COBOL Stored Procedure Builder.

1. Copy ELAXMSAM, to the DB2 Proclib and customize the JCL as described in its comment. Make sure that the SYSEXEC DD card points to the library where the REXX code of the PL/I and COBOL Stored Procedure Builder resides.
2. Use the workload management panels to associate an application environment with the JCL procedure of the WLM address space for the PL/I and COBOL Stored Procedure Builder. See OS/390 MVS Planning: Workload Management for information on how to do this.
3. Copy ELAXMJCL to a private JCL library, customize as described in its comments and execute the job. Make sure the WLM ENVIRONMENT clause in the CREATE PROCEDURE statement specifies the name of the WLM environment which has been defined for the PL/I and COBOL Stored Procedure Builder.

Customizing the Enterprise Developer/SCLM Support

If you plan on using SCLM as a version control manager with WebSphere Studio Enterprise Developer, Enterprise Developer Options for z/OS provides the necessary code for the interface. You will need to customize this code in order to enable this interface. Additional documentation on the Enterprise Developer/SCLM interface can be found in the HEDS500/SCLMDOCS directory on the install CD-ROM.

Verifying the install of the modules: After the SMP/E apply step, the Enterprise Developer/SCLM z/OS components should reside in a PDS loadlib named hlq.CCU.V5R0M0.SCCULOAD and two modules should reside in the hfs /usr/lpp/EnterpriseDeveloper/SCLM directory (default from the Enterprise Developer Options for z/OS install) 18

The two hfs modules will be :

- ELAXCALL
- ELAXTSO

The module ELAXTSO should have the sticky bit turned on already. To verify, an LS -E ELAXTSO should display the module attributes as such rwxr_xr_xt.

The load library hlq.CCU.V5R0M0.SCCULOAD should already be APF authorized. See Section “Making the Enterprise Developer Options for z/OS Load Library APF authorized” on page 4

ISPF Configuration File The ISPF configuration table ISPF.conf will need to be customized to Host Site requirements for ISPF dataset allocation by the system programmer after WSED/SCLM has been installed. It is recommended to copy and rename the supplied sample configuration file ELAXWEBI to /etc/EnterpriseDeveloper/SCLM/ISPF.conf. If you use a different path you will

need to update CGI_ELAXPATH webserver environment variable . The sample HTTP webserver environment variable file will have by default : CGI_ELAXPATH = /etc/EnterpriseDeveloper/SCLM

Note: If the environment variable CGI_ELAXPATH is not included the program will look for the ISPF.conf file in the WSED/SCLM install path. (By Default: /usr/lpp/EnterpriseDeveloper/SCLM)

The provided sample ISPF.conf (ELAXWEBI) has instructions to complete customization and enables user site to :

- Include the minimum ISPF dataset allocations for SCLM plugin operation
- Add additional DDNAME file allocations or concatenate additional ISPF datasets
- Execute a customer defined allocation exec to provide further dataset allocation by Project or Userid

Configuring the Enterprise Developer/SCLM Webserver: The Enterprise Developer/SCLM interface uses a web server to communicate with the Enterprise Developer workstation client. This section will describe the setup and customization of a IBM HTTP webserver that will be required by Enterprise Developer to access SCLM on a z/OS host. It is recommended that this be a dedicated web server to support this interface though optionally you may incorporate the required SCLM /HTTP configuration directives into an existing HTTP server. See “Customizing an existing HTTP server for SCLM support” on page 22. By default the SCLM/HTTP server will be configured to use port 80 though a suitable dedicated port may be chosen during customization. If you change the default port number it must be changed in the server JCL and in the elaxwebc.conf file.

The sample setup will require the end user to supply a valid z/OS userid and password when accessing the host system using this interface.

Note: For additional information on configuring IBM HTTP web servers, review the HTTP Server Planning, Installing, and Using manual for your level of the operating system.

The following sections outline the steps for customizing the supplied samples.

HTTP Server Configuration and Environment File Customization for SCLM

1. (Optional, but recommended) Create a directory for the customized HTTP Server configuration file and the environment file. The permissions should be set to 755 and should be read only to make sure the files cannot be changed except by authorized people. In this directory, create a ″logs″ and a ″reports″ directory. The permissions for these directories should also be set to 755. The resulting example directories should be:

   ```
   /u/vcm
   /u/vcm/logs
   /u/vcm/reports
   ```

   The directory path /u/vcm is just a sample directory path given for the required http servers configuration files and logs. You may determine your own http server directory and reflect those changes accordingly.
2. Copy the sample HTTP configuration file elaxwebc.conf from the install directory /usr/lpp/EnterpriseDeveloper/SCLM/ into the created directory (/u/vcm/). Follow the instructions in the configuration file for the changes that are needed.

3. Copy the sample HTTP environment variables file elaxwebe.envvars from the install directory /usr/lpp/EnterpriseDeveloper/SCLM/ into the created directory (/u/vcm/). Follow the instructions in the environment file for the changes that are needed.
4. The default port to be used is 80. If you change this to a specific dedicated port then reflect this port number both in the started task jcl for the webserver and the elaxwebc.conf configuration file.
5. **Non-standard codepage translation in WSED/SCLM:** If users require different ASCII/EBCDIC codepage translation other than standard default (IBM-1047 / ISO8859-1) the following parameters must be coded in the httpd.conf (elaxwebc.conf) file for the HTTP webserver.

```
 DefaultFsCp ebcdic-codepage
DefaultNetCp ascii-codepage
```

For Japanese translation the required codepages would be:

```
DefaultFsCp IBM-939
DefaultNetCp IBM-932C
```

Notes®:

Note: The following restrictions exist with Windows® Japanese in regards to SCLM host character translation. The following EBCDIC characters translate to ? (0x3F) on WSED. pond (0xB1) cent (0x4A) Not (0x5F).

HTTP Webserver JCL / STARTED TASK customization for SCLM

1. Copy the sample batch job ELAXWEBJ (see the sample below) from the installed sample library, hlq.CCU.V5R0M0.SCCUSAMP, to a JCL library or PROCLIB dataset and customize to your site specific standards and information.
2. It is recommended to make this HTTP server job a started task.
3. Replace the directory ″/u/vcm″ in the JCL to the directory containing the customized configuration file (default name: elaxwebc.conf) and environment variables file (default name: elaxwebe.envvars).
4. The default port to be used is 80. If you change this to a specific dedicated port you must change the port number in 2 places:
   - In the started task JCL for the webserver 20
   - In the elaxwebc.conf configuration file
5. Edit the STEPLIB to specify the APF authorized loadlib containing the WebSphere Studio Enterprise Developer Option for z/OS loadlib members. The default install directory is hlq.CCU.V5R0M0.SCCULOAD.

   Alternatively the loadlib dataset could be included in the Linklist.

```
Alternatively the loadlib dataset could be included in the Linklist.
//VCMWEBJ JOB (accnt),'VCM',NOTIFY=&SYSUID,MSGCLASS=X,
//   TIME=(5),CLASS=A,MSGLEVEL=(1,1),REGION=128M
//*
//* (or create as started task )
//*
//*VCMWEBJ PROC
//*
//*-------------------------------------------------------------
//VCMWEB EXEC PGM=IMWHTTPD,TIME=NOLIMIT,ACCT=(ACCT#),
//   PARM=('ENVAR("_CEE_ENVFILE=/u/vcm/elaxwebe.envvars")/-r
//        /u/vcm/elaxwebc.conf -B -p 8')
//*
//STEPLIB   DD DSN=hlq.CCU.V5RM.SCCULOAD,DISP=SHR
//*
```

```
//SYSIN     DD DUMMY
//OUTDSC    OUTPUT DEST=HOLD
//SYSPRINT  DD SYSOUT=,OUTPUT=(*.OUTDSC)
//SYSTSPRT  DD SYSOUT=,OUTPUT=(*.OUTDSC)
//STDOUT    DD SYSOUT=,OUTPUT=(*.OUTDSC)
//STDERR    DD SYSOUT=,OUTPUT=(*.OUTDSC)
//SYSOUT    DD SYSOUT=,OUTPUT=(*.OUTDSC)
//SYSERR    DD SYSOUT=,OUTPUT=(*.OUTDSC)
//CEEDUMP   DD SYSOUT=,OUTPUT=(*.OUTDSC)
//
```

6. Either the job or the started task needs to be authorized to execute the HTTP Server modules. This can be done by associating the job or started task to the IMWEB group and the WEBSRV userid.

   The following information was taken from the HTTP Planning, Installing, and Using Manual, IBM document number SC31-8903.

   - For the started task to obtain control with the desired user identity, you must add an entry to the RACF started procedures table, module ICHRIN03. This entry defines the user ID and group ID that the IMWEBSRV address space will be assigned.
   - To assign the user ID, WEBSRV, to the RACF started procedures table, module ICHRIN03, see the following example (the example assumes you have used VCMWEBJ as the started procedure name):

     ```
     DC CL8'VCMWEBJ' PROCEDURE NAME
     DC CL8'WEBSRV' USERID (ANY RACF-DEFINED USER ID)
     DC CL8'IMWEB' GROUP NAME/BLANKS FOR USER'S DEFAULT GROUP
     DC XL1'00' NOT TRUSTED
     DC XL7'00' RESERVED
     ```

     Note: Remember to increment the count of defined started procedures.
   - Or, you can add the PROC to the started task table. The following example provides a way to add the server PROC to the started task table using RACF commands:
     - RALTER STARTED VCMWEBJ.** STDATA(USER(WEBSRV))
     - SETROPTS RACLIST(STARTED) REFRESH
   - If you are defining the IMWEBSRV.** profile for the first time, use RDEFINE statements instead of RALTER statements.

7. Ensure a REXX/370 runtime environment on the host exists or alternatively use the REXX/370 Alternate Library.

   The REXX runtime library REXX.**.SEAGLPA (or if not installed, the alternate library REXX.**.SEAGALT) will need to be added to the STEPLIB in the HTTP webserver's JCL if they are not already defined as LNKLST datasets.

   These datasets will need to be APF authorized.

Customizing an existing HTTP server for SCLM support

Follow the instructions below if you optionally choose to incorporate the SCLM support into an existing HTTP web server.

1. Add the following pass/exec directives in the httpd.conf configuration file.

   ```
   Pass /elaxtest.html /usr/lpp/EnterpriseDeveloper/SCLM/elaxtest.html
   Exec /ELAXCALL /usr/lpp/EnterpriseDeveloper/SCLM/ELAXCALL
   ```

2. Ensure the installed loadlib hlq.CCU.V5R0M0.SCCULOAD is either in the linklist or included in the STEPLIB for the HTTP server.

Start the Enterprise Developer/SCLM HTTP webserver: Start the web server by submitting the job or if a started task procedure enter the following command from the z/OS console:

- S VCMWEBJ

(Ensure the procedure is a member in a PROCLIB data set).

Testing the HTTP webserver: From a browser, type the location URL address: http://hostname:portnumber/elaxtest.html

Hostname is the TCP/IP host name the HTTP Server is running on and the port number is the port used in the job and the elaxwebc.conf file (default port 80).

You will prompted for a valid userid and password for the system the web server is started on.

The browser should then display the message:

```
z/OS Host Connection Successful
```

Customizing for EGL support

The following customization information only applies of you are planning on using EGL (FMID: H284500)

Customizing the Remote Build Servers

Enterprise Developer Options for z/OS provides code for z/OS and USS remote build servers. These servers are optional depending on the functions you are using in Enterprise Developer.

The z/OS server enables you to remotely build COBOL, PL/I, and EGL generated COBOL programs on z/OS host systems. The Unix System Services (USS) server enables you to remotely build Java programs on USS host systems.

If either of these build servers are to be used, it is required that you configure and start the server. If you need both build servers, then you need to start two servers, each listening on a unique TCP/IP port for each type.

The Remote Build server performs the following tasks:

- Receives build requests and files
- Performs character conversions
- Runs builds within its environment
- Optionally collects and returns results to the client

Customizing the z/OS Remote Build Server: This section describes the steps to configure and start the build server needed to build COBOL, PL/I, and EGL generated COBOL programs on z/OS. After editing the JCL, ensure this server is started so it is available at all times for the developers.

Note: 5.1.2 Clients require the build server for COBOL and PL/I generated programs. The build server is not needed for manually developed programs.

The build server for z/OS is started as an z/OS job or started task. Sample JCL to run this job is provided in the CCURUNU member of the hlq.CCU.V5R0M0.SCCUSAMP (default name) sample library. This member should be updated to reflect any of your site standards. It can be copied to the system PROCLIBs if you want to start this as a started task.

The CCURUNU job starts the main server. When a build request is received from a client, a new build transaction job is started on z/OS to perform the actual builds. The job that is started is specified in the CCUWJCL DDname in the CCURUNU sample job. Sample JCL for this new job on z/OS is in the member of hlq.CCU.V5R0M0.SCCUSAMP. The server is multi-threaded, so these jobs run concurrently and are independent of each other. The number of concurrent jobs running at any one time is limited by system resources (such as initiators).

It is recommended that these jobs be copied to some other data set so that when maintenance is applied, modifications will not be lost.

The sample CCURUNU job looks like this:

```
//JOBNAME JOB (ACCT#),'TSO ID',CLASS=A
//RUNPGM EXEC PGM=CCUMAIN,REGION=7400K,
// PARM='-p 5555 -V -a 0 -n 2 -q 10'
//STEPLIB DD DSN=hlq.CCU.V5R0M0.SCCULOAD,DISP=SHR
//CCUWJCL DD DISP=SHR,DSN=hlq.CCU.V5R0M0.SCCUSAMP(CCUUSS)
//STDOUT DD SYSOUT=*
//STDERR DD SYSOUT=*
//CCUBLOG DD SYSOUT=*
//
```

Edit CCURUNU as follows:

1. Replace the first line with a valid job card. Since this will most likely be a longrunning server, TIME=NOLIMIT is recommended.
2. Edit the parameters for the PARM as follows. All are optional except for -p The parameters are:

   -p Port number listened on. Must be a unique TCP/IP port number.

   -V Specifies the verbosity level of the server. It is case sensitive. You may specify this parameter up to three times (maximum verbosity).

   For example, to increase the verbosity to the maximum, you specify -V -V -V

   -a Specifies the authentication mode of the build server.

   2 The user submitting the build request must specify a valid TSO user ID and password when the user initiates a build by using the remote build client. The server performs the new build transaction under the access authority of this user ID.

   1 The user submitting the build request can provide a valid TSO user ID and password. If one is provided, the server performs the new build transaction under the access and authority of the provided user. If the user does not provide a user ID and password, the new build transaction is performed under the access authority of the user ID assigned to the build server job.

   0 The new build transaction is always performed under the access authority of the user ID assigned to the build server job. If the build server is not started from an APF authorized library, any requests by the new build transaction that execute programs from an APF authorized library will fail. Mode 0 is the default.

   You can use modes 1 and 2 only if the server load modules are run from an APF authorized library. If you start the server from an APF authorized library (required in modes 1 and 2 but optional in mode 0),

the new build transaction can specify an APF authorized program as the executable. The build script can also execute non-APF authorized programs. However, in a multistep JCL script, an authorized program cannot be executed after an unauthorized program.

If the server is not started from an APF authorized library (allowed in mode 0), the build script can only execute non-APF authorized programs.

If the CICS® translator libraries and the COBOL compiler libraries are not APF authorized, when running in authentication mode 1 or 2, the build job will receive an ABEND 306. The build server will recover from this abend and complete successfully. You can prevent the 306 abend by making the libraries containing programs invoked by the build server APF authorized.

-n Specifies the number of concurrent builds. This determines how many CCUUSS jobs can run at the same time. Default is 2. Once the specified number of concurrent builds are running, the build server queues any additional requests and submits them on a first come, first served basis as builds are completed. The number of initiators for the CCURUNU job class should also equal the value of -n.

-q Specifies the queue size for client requests. The default is 10. Each queued client uses a TCP/IP socket. Therefore setting this too high may require more sockets than are available, causing unpredictable results. If the queue is full, subsequent clients are rejected by the server. However, the build client will automatically retry the build 10 times.

-t Used to turn trace on for diagnostic purposes. The trace will be written to the file specified in the CCUBLOG DD card. This parameter should only be used under direction of the IBM support center. (Parameter not shown in sample JCL)

3. Edit the STEPLIB so that it specifies the name of the library containing the Enterprise Developer Options for z/OS load modules (default is hlq.CCU.V5R0M0.SCCULOAD). This library must be APF authorized if running in authentication mode 1 or 2.
4. Edit the CCUWJCL DD to specify the name of the member containing the sample JCL used when submitting the build. hlq.CCU.V5R0M0.SCCUSAMP(CCUUSS) contains the sample JCL.

The sample CCUUSS job looks like this:

```
//JOBNAME JOB (ACCT#),'TSO ID',CLASS=A
//RUNPGM EXEC PGM=CCUBLDW,DYNAMNBR=30,TIME=NOLIMIT,
// PARM='ENVAR("_CEE_ENVFILE=DD:EDCENV")/
// '&PARM'
//STEPLIB DD DSN=hlq.CCU.V5R0M0.SCCULOAD,DISP=SHR
//CCUPROC DD DSN=hlq.CCU.V5R0M0.SCCUSAMP,DISP=SHR
//STDOUT DD SYSOUT=*
//STDERR DD SYSOUT=*
//CCUBLOG DD SYSOUT=*
//EDCENV DD DUMMY
//EDCENV DD DSN=hlq.SOME.SEQ.FILE,DISP=SHR
//
```

Edit the CCUUSS member as follows:

1. Replace the first line with a valid job card. The jobname must be less than 8 characters. The jobs will be submitted by appending a number to jobname above. For example JOBNAME0, JOBNAME1 etc. in this example. There should

be at least 3 blanks before the JOB keyword. When the build server is started with authorization mode 2, &USERID may be specified for the job name which will cause the job name to be the user ID passed from the client appended with a number. This feature may only be used if user IDs are restricted to less than 8 characters.

2. Avoid changing the PARM statement. The &PARM keyword will be replaced by parameters sent from the main build server component.
3. Edit the STEPLIB so that it specifies the name of the library containing the Enterprise Developer Options for z/OS load modules (default is hlq.CCU.V5R0M0.SCCULOAD). If this library is APF authorized, the build server can run authorized programs specified in the build scripts (such as IEBCOPY). Otherwise, only non-authorized programs are allowed.
4. Modify the DSN name in the CCUPROC DDname so that it points to the PDS that contains pseudo-JCL build scripts. When building EGL generated COBOL programs, this DSname needs to specify the data set containing the customized EGL JCL build scripts. See "Customizing the EGL JCL Build Scripts" on page 15 for more information on these build scripts.
5. The EDCENV input data set is used to provide optional variables. It can be set to DUMMY if there are no options to be set. The only optional variable that can be set is for tracing.

   To turn tracing on:

   - Create a sequential file with a Fixed Block (FB) record format and a record length (LRECL) of 80. Replace the hlq.SOME.SEQ.FILE in the sample JCL with the name of this dataset and uncomment out the line.
   - Comment out the line that associates EDCENV with DUMMY
   - Add the following line to the sequential dataset.

     ```
     CCU_TRACE=*
     ```

   - The trace will be written to CCUBLOG.

   To turn tracing off, edit the JCL so the EDCENV DD is set to DUMMY or modify the line in the sequential file to add an initial "*". The resulting line would be as follows:

   ```
   *CCU_TRACE=*
   ```

   Tracing will cause performance degradations and should only be done under the direction of the IBM support center.

Note: If the build server is

- Started on a machine with multiple partitions, and
- A job submitted in one partition may be started in another partition

Then you need to customize both the server start up JCL(CCURUNM or CCURUNU) and the build transaction JCL(CCUMVS or CCUUSS) to force the jobs to run in the same partition.

For JES2 systems this can be accomplished using the JOBPARM control statement. For example:

```
/*JOBPARM SYSAFF=SYS1
```

Forces a job to be run in the partition named SYS1.

For JES3 systems this can be accomplished using the MAIN control statement. For example:

```
//*MAIN SYSTEM=(SYS1)
```

Forces a job to be run in the partition named SYS1.

Starting the Build Server for z/OS

Submit the job contained in the CCURUNU member.

Stopping the Build Server for z/OS

Cancel the job that was used to start it.

Verifying the installation of the z/OS Build server.

A build client is provided in two places:

1. On the Host installation CD-ROM
2. On the Client installation CD-ROM

Select the build client you wish to use to verify the installation and customization of the USS Build server.

1. To perform verification using the build client that is located on the Host installation CD-ROM do these steps:
   a. Ensure the dataset specified in the CCUPROC DD statement contains the member CCUTRUE. The CCUTRUE member is shipped in the hlq.CCU.V5R0M0.SCCUSAMP dataset.
   b. Start the z/OS build server (as a job or as a started task).
   c. On a workstation, open a command prompt window and execute one of the following commands:
      - When the build server is started in Authentication mode 2 or 1

        ```
        d:\HEDS500\ccubldc -h hostname@port -b true -au USERID -ap PASSWORD
        ```

      - When the build server is started in Authentication mode 0

        ```
        d:\HEDS500\ccubldc -h hostname@port -b true
        ```

        Where:

        d: is the CD-ROM drive for your workstation.

        hostname
        is the TCP/IP name of the host machine where the build server is running.

        port is the port number in the -p parameter of the USS build server job.

        true is the name of the command to run in the USS build script. This is not to be changed.

        USERID
        is the userid the new build transaction will run with. This must be entered in upper case.

        PASSWORD
        is the password the new build transaction will run use. This must be entered in upper case. It is not required if running in authentication mode 1.

d. The following should be returned if successful.

```
02/09/05 19:52:51 (c) Copyright, IBM Corp. 2001
        Copyright (c) 2002 Rational Software Corporation
02/09/05 19:52:57 *** Success ***
02/09/05 19:52:57
CCUI-003
Program Name : 'IEFBR14 '.
PARM :         '&PARM'.

CCUI-004
The z/OS step 'T2' return code is '000000'.
02/09/05 19:52:57 *---------------------------------
```

2. To use the build client provided on the Client install CD-ROM to verify the installation and customization of the z/OS Build server, perform the following steps:
   a. Ensure the dataset specified in the CCUPROC DD statement contains the member CCUTRUE. The CCUTRUE member is shipped in the hlq.CCU.V5R0M0.SCCUSAMP dataset.
   b. Start the z/OS build server (as a job or a started task).
   c. On a workstation, open a command prompt window and execute one of the following commands:
      - When the build server is started in Authentication mode 2 or 1

        ```
        x:<WSED_install>ccubldc -h hostname@port -b
                                    ccutrue -au USERID -ap PASSWORD
        ```

      - When the build server is started in Authentication mode 0

        ```
        x:\<WDz_install>\bin\ccubldc -h hostname@port -b ccutrue
        ```

        Where:

        x:\<WDz_install>
        is the drive and directory where WSED is installed (″C:\Program Files\IBM\Rational\SDP\6.0″) is the default.

        hostname
        is the TCP/IP name of the host machine where the build server is running.

        port is the port number in the -p parameter of the z/OS build server job.

        ccutrue
        is the name of the proc to run in the new build transaction. This is not to be changed.

        USERID
        is the userid the new build transaction will run with. This must be entered in upper case.

        PASSWORD
        is the password the new build transaction will run use. This must be entered in upper case. It is not required if running in authentication mode 1.

   d. The following should be returned if successful.

```
02/09/05 19:52:51 (c) Copyright, IBM Corp. 2001
         Copyright (c) 2002 Rational Software Corporation
02/09/05 19:52:57 *** Success ***
02/09/05 19:52:57
CCUI-003
Program Name : 'IEFBR14 '.
```

```
PARM :         '&PARM'.

CUI-004
The z/OS step 'T2' return code is '000000'.
02/09/05 19:52:57 *---------------------------------
```

Customizing the USS Remote Build Server:

This section describes the steps to configure and start the build server needed to build EGL generated Java programs on Unix System Services (USS). After editing the JCL, ensure this server is started so it is available at all times for the developers.

The build server for USS is started as an z/OS job or started task. Sample JCL to run this job is provided in the CCURUNU member of the hlq.CCU.V5R0M0.SCCUSAMP (default name) sample library. This member should be updated to reflect any of your site standards. It can be copied to the system PROCLIBs if you want to run this as a started task.

The CCURUNU job starts the main server. When a build request is received from a client, a new z/OS build transaction job is started for USS builds. This new job then executes a shell script under USS (using BPXBATCH) to actually perform the build. The job that is started is specified in the CCUWJCL ddname in the CCURUNU sample job. Sample JCL for this new job on USS is in the CCUUSS member of hlq.CCU.V5R0M0.SCCUSAMP.

The server is multi-threaded, so these jobs run concurrently and are independent of each other. The number of concurrent jobs running at any one time is limited by system resources (such as initiators).

It is recommended that these jobs be copied to some other data set so that when maintenance is applied, modifications will not be lost.

The sample CCURUNU job looks like this:

```
//JOBNAME JOB (ACCT#),'TSO ID',CLASS=A
//RUNPGM EXEC PGM=CCUMAIN,REGION=7400,
// PARM='-p 5555 -V -a 0 -n 2 -q 10'
//STEPLIB DD DSN=hlq.CCU.V5R0M0.SCCULOAD,DISP=SHR
//CCUWJCL DD DISP=SHR,DSN=hlq.CCU.V5R0M0.SCCUSAMP(CCUUSS)
//STDOUT  DD SYSOUT=*
//STDERR  DD SYSOUT=*
//CCUBLOG DD SYSOUT=*
//
```

Edit CCURUNU as follows:

1. Replace the first line with a valid job card. Since this will most likely be a longrunning server, TIME=NOLIMIT is recommended.
2. Edit the parameters for the PARM as follows. All are optional except for -p The parameters are:

   -p Port number listened on. Must be a unique TCP/IP port number.

   -V Specifies the verbosity level of the server. It is case sensitive. You may specify this parameter up to three times (maximum verbosity).

   For example, to increase the verbosity to the maximum, you specify -V -V -V

   -a Specifies the authentication mode of the build server.

2 The user submitting the build request must specify a valid TSO user ID and password when the user initiates a build by using the remote build client. The server performs the new build transaction under the access authority of this user ID.

1 The user submitting the build request can provide a valid TSO user ID and password. If one is provided, the server performs the new build transaction under the access and authority of the provided user. If the user does not provide a user ID and password, the new build transaction is performed under the access authority of the user ID assigned to the build server job.

0 The new build transaction is always performed under the access authority of the user ID assigned to the build server job. If the build server is not started from an APF authorized library, any requests by the new build transaction that execute programs from an APF authorized library will fail. Mode 0 is the default.

You can use modes 1 and 2 only if the server load modules are run from an APF authorized library. If you start the server from an APF authorized library (required in modes 1 and 2 but optional in mode 0), the new build transaction can specify an APF authorized program as the executable. The build script can also execute non-APF authorized programs. However, in a multistep JCL script, an authorized program cannot be executed after an unauthorized program.

If the server is not started from an APF authorized library (allowed in mode 0), the build script can only execute non-APF authorized programs.

If the CICS translator libraries and the COBOL compiler libraries are not APF authorized, when running in authentication mode 1 or 2, the build job will receive an ABEND 306. The build server will recover from this abend and complete successfully. You can prevent the 306 abend by making the libraries containing programs invoked by the build server APF authorized.

-n Specifies the number of concurrent builds. This determines how many CCUUSS jobs can run at the same time. Default is 2. Once the specified number of concurrent builds are running, the build server queues any additional requests and submits them on a first come, first served basis as builds are completed. The number of initiators for the CCURUNU job class should also equal the value of -n.

-q Specifies the queue size for client requests. The default is 10. Each queued client uses a TCP/IP socket. Therefore setting this too high may require more sockets than are available, causing unpredictable results. If the queue is full, subsequent clients are rejected by the server. However, the build client will automatically retry the build 10 times.

-t Used to turn trace on for diagnostic purposes. The trace will be written to the file specified in the CCUBLOG DD card. This parameter should only be used under direction of the IBM support center. (Parameter not shown in sample JCL)

3. Edit the STEPLIB so that it specifies the name of the library containing the Enterprise Developer Options for z/OS load modules (default is hlq.CCU.V5R0M0.SCCULOAD). This library must be APF authorized if running in authentication mode 1 or 2.

4. Edit the CCUWJCL DD to specify the name of the member containing the sample JCL used when submitting the build. hlq.CCU.V5R0M0.SCCUSAMP(CCUUSS) contains the sample JCL.

The sample CCUUSS job looks like this:

```
//JOBNAME JOB (ACCT#),'TSO ID',CLASS=A
//RUNPGM EXEC PGM=BPXBATCH,DYNAMNBR=30,REGION=7400K,
//     PARM='SH /usr/lpp/EnterpriseDeveloper/BuildServer/ccubldw.sh
//              '&PARM'
//
```

Edit the CCUUSS member as follows:

1. Replace the first line with a valid job card. The jobname must be less than 8 characters. The jobs will be submitted by appending a number to jobname above. For example JOBNAME0, JOBNAME1 etc. in this example. There should be at least 3 blanks before the JOB keyword. When the build server is started with authorization mode 2, &USERID may be specified for the job name which will cause the job name to be the user ID passed from the client appended with a number. This feature may only be used if user IDs are restricted to less than 8 characters.
2. Be careful changing the PARM statement. The &PARM keyword will be replaced by parameters sent from the main build server component. This should remain in column 16 if it remains a continuation line.
3. Specify the location of the shell script that will be started by this job. The default shell script shipped with Enterprise Developer Options for z/OS is "ccubldw.sh" and is located in the path /usr/lpp/EnterpriseDeveloper/BuildServer. If this script has been moved to some other path, then change the path in the PARM statement. (For example, if you specified a path prefix in the job CCUAMKD).

The default shell script ccubldw.sh invoked by CCUUSS contains the following:

```
export PATH=/usr/lpp/EnterpriseDeveloper/BuildServer:$PATH
export CLASSPATH=/usr/lpp/EnterpriseDeveloper/EGLJava/fda.jar:
          /usr/lpp/EnterpriseDeveloper/EGLJava/fdaj.jar:$CLASSPATH
export LIBPATH=/usr/lpp/EnterpriseDeveloper/BuildServer:.:$LIBPATH
#export CCU_TRACE=
#export CCU_TRACEFILE=/u/userid/ccu.trace
/usr/lpp/EnterpriseDeveloper/BuildServer/ccubldw $1 $2 $3 $4 $5
echo $1 $2 $3 $4 $5
echo $?
```

Edit the shell script as follows:

Note: If you specified a path prefix in the job CCUAMKD, the default paths described below will need this path prefix added to the beginning of the path.

1. Edit the PATH statement to specify the location of the Enterprise Developer Options for z/OS executables. The default location is /usr/lpp/EnterpriseDeveloper/BuildServer.
2. Edit the CLASSPATH to specify the location of the Enterprise Generation Language runtime Jar files fda.jar and fdaj.jar. The default location is: /usr/lpp/EnterpriseDeveloper/EGLJava. Also, if needed, add any additional jar files or directories needed to compile non-EGL Java source. In the sample

ccubldw.sh the CLASSPATH specification is on one line. If you are not going to build EGL generated Java programs, then you can remove the references to the fda.jar and fdaj.jar files.

3. Edit the LIBPATH statement to specify the location of the Enterprise Developer Options for z/OS executables. The default location is /usr/lpp/EnterpriseDeveloper/BuildServer.
4. Turn on tracing if desired. This can be done by uncommenting the lines for CCU_TRACE=* and CCU_TRACEFILE=/ccu.trace. For example:

   ```
   export CCU_TRACE=
   export CCU_TRACEFILE=/u/userid/ccu.trace
   ```

   /u/userid should be replaced with the appropriate home directory that the output should be written to. If /u/userid is replaced with the "tilde" character, the file will be written to the home directory of userid that the CCUUSS job was running under.

   Tracing will cause performance degradation and should only be done under the direction of the IBM support center
5. Edit /usr/lpp/EnterpriseDeveloper/BuildServer/ccubldw to specify the path of where the ccubldw executable is located. /usr/lpp/EnterpriseDeveloper/BuildServer is the default path.

Note: If the build server is

- Started on a machine with multiple partitions, and
- A job submitted in one partition may be started in another partition

Then you need to customize both the server start up JCL(CCURUNM or CCURUNU) and the build transaction JCL(CCUMVS or CCUUSS) to force the jobs to run in the same partition.

For JES2 systems this can be accomplished using the JOBPARM control statement. For example:

```
/JOBPARM SYSAFF=SYS1
```

Forces a job to be run in the partition named SYS1.

For JES3 systems this can be accomplished using the MAIN control statement. For example:

```
//MAIN SYSTEM=(SYS1)
```

Forces a job to be run in the partition named SYS1.

Starting the Build Server for USS:

Submit the job contained in the CCURUNU member.

Stopping the Build Server for USS:

Cancel the job that was used to start it.

Verifying the installation of the USS Build server.

A build client is provided on the install CD-ROM to enable the verification of the installation and customization of the USS build server. To perform this verification, perform the following steps:

1. Start the USS build server (as a job or as a started task).
2. On a workstation, open a command prompt window and execute one of the following commands:
   - When the build server is started in Authentication mode 2 or 1

     ```
     d:\HEDS500\ccubldc -h hostname@port -b true -au USERID -ap PASSWORD
     ```

     When the build server is started in Authentication mode 0

     ```
     d:\HEDS500\ccubldc -h hostname@port -b true
     ```

   Where:

   d: is the CD-ROM drive for your workstation.

   hostname
   is the TCP/IP name of the host machine where the build server is running.

   port is the port number in the -p parameter of the USS build server job.

   true is the name of the command to run in the USS build script. This is not to be changed.

   USERID
   is the userid the new build transaction will run with. This must be entered in upper case.

   PASSWORD
   is the password the new build transaction will run use. This must be entered in upper case. It is not required if running in authentication mode 1.
3. The following should be returned if successful.

```
02/09/05 20:47:23 (c) Copyright, IBM Corp. 2001
       Copyright (c) 2002 Rational Software Corporation
02/09/05 20:47:34 *** Success ***
02/09/05 20:47:34
Command: true
****************** Build Script Output Follows **********
****************** End Of Build Script Output ***********
02/09/05 20:47:35 *-------------------------------------
```

To perform verification using the build client that is located on the Client installation CD-ROM do these steps:

1. Start the USS build server (as a job or a started task).
2. On a workstation, open a command prompt window and execute one of the following commands:
   - When the build server is started in Authentication mode 2 or 1

     ```
     d:\HEDS500\ccubldc -h hostname@port -b true -au USERID -ap PASSWORD
     ```

   - When the build server is started in Authentication mode 0

     ```
     d:\HEDS500\ccubldc -h hostname@port -b true
     ```

   Where:

   d: is the CD-ROM drive for your workstation.

   hostname
   is the TCP/IP name of the host machine where the build server is running.

   port is the port number in the -p parameter of the USS build server job.

true is the name of the command to run in the USS build script. This is not to be changed.

USERID
is the userid the new build transaction will run with. This must be entered in upper case.

PASSWORD
is the password the new build transaction will run use. This must be entered in upper case. It is not required if running in authentication mode 1.

3. The following should be returned if successful.

```
02/09/05 2:47:23 (c) Copyright, IBM Corp. 2001
          Copyright (c) 2002 Rational Software Corporation
02/09/05 20:47:34 *** Success ***
02/09/05 20:47:34
Command: true
****************** Build Script Output Follows **********
****************** End Of Build Script Output ***********
2/9/5 2:47:35 *-------------------------------------
```

Customizing the EGL JCL Build Scripts

Enterprise Developer Options for z/OS provides JCL build scripts to be used for building EGL generated COBOL programs. A build script is a file used by the build server to transform one set of files into another. For example, the z/OS build server uses a build script written in pseudo-JCL to transform a COBOL source file into an object file and (in some cases) to transform one or more object files into a load module.

When EGL generates COBOL for an z/OS environment, the generator specifies predefined pseudo-JCL build scripts based on the program being generated. There is a build script to invoke the z/OS COBOL compiler and the z/OS linker. If the generated program contains CICS commands, a build script is used that calls the CICS translator as well. If the generated program contains SQL, build scripts are provided that invoke the DB2 preprocessor and perform a DB2 Bind.

The pre-defined EGL build scripts are provided with Enterprise Developer Options for z/OS. They are shipped in the default library name hlq.CCU.V5R0M0.SCCUSAMP. The job CCURUNU contains a DD card CCUPROC that specifies the location of these build scripts. See “Customizing the z/OS Remote Build Server” on page 4 for more information.

The pre-defined EGL JCL build scripts are as follows:

FDACL
Compile and link of EGL generated COBOL programs that do not contain CICS commands or DB2 statements.

FDATCL
CICS Translation, compile, and link of EGL generated CICS COBOL programs that do not contain DB2 statements.

FDAPTCL
DB2 Precompile, CICS Translation, compile, and link of generated CICS COBOL programs that do contain DB2 Statements.

FDABIND
Script to perform bind for generated programs that contain DB2 statements.

FDALINK
Links objects mofules produced from compiling EGL generated COBOL programs containing DB2 statements.

FDAPCL
DB2 precompile, compile and link of EGL generated COBOL programs that do not contain CICS commands.

FDABCL
Comile and link of batch EGL programs that do not contain CICS commands or DB2 statements.

FDAMFS
Assemble and link MFS source generated from EGL programs for IMS/VS.

A Enterprise Developer developer can override the use of the pre-defined build scripts by using a EGL build descriptor symbolic parameter named DISTBUILD_BUILD_SCRIPT. See the EGL on-line helps in Enterprise Developer for more information on how to use symbolic parameters during generation.

These build scripts are developed using pseudo-JCL which is a form of JCL with some restrictions and a few extensions. The build server reads the pseudo-JCL and processes it to invoke the build program for the data sets specified for the program. Because the build scripts reference data sets for the DB2 preprocessor, the CICS translator, the COBOL compiler, and the z/OS linkage editor, you need to modify the shipped build scripts to at least specify the actual locations of these components.

For a description on how to write or modify a JCL build script, the pseudo-JCL syntax, pre-defined pseudo-JCL substitution variables, and pre-defined symbolic parameters for EGL generation, see either the on-line helps for EGL within Enterprise Developer or the IBM Enterprise Developer Server Guide for z/OS.

Note: A number of the substitution variables are provided by the build client (on the workstation) when it invokes the build server.

To perform the minimal set of changes to the build scripts, complete the following changes.

1. Edit all sample scripts by:
   - Copying the members to some other dataset before editing. (Recommended so that when maintenance is applied, changes will not be lost).
   - Editing the members in the Enterprise Developer Options for z/OS default library hlq.CCU.V5R0M0.SCCUSAMP.
2. Replace the dataset names in the build scripts for the following substitution variables (called ″VARS″) to reflect the names used in your installation. The example shows how the substitution variables are specified.

```
//*************************************************************
//* FDAPTCL - DB2 PRECOMPILE, CICS TRANSLATOR, COBOL COMPILE,
//* AND LINK-EDIT
//*************************************************************
//*
//DEFAULTS VARS CGHLQ=USER,
// COBCICS=CEE.SCEECICS,
// COBCOMP=IGY.SIGYCOMP,
// COBLIB=CEE.SCEELKED,
// COBLISTPARMS=OFFSET &COMMA.NOLIST&COMMA.MAP,
// ELA=ELA.V5R0M0,
// DATA=31,
```

```
// DFHLOAD=DFH.SDFHLOAD,
// DSNEXIT=DSN.DSNEXIT,
// DSNLOAD=DSN.DSNLOAD,
// SYSTEM=MVSCICS,
// MBR=TEMPNAME,
// RGN=4096K,
// CCUEXTP=CCUOUT,
```

COBCICS
The name of the Language Environment® (LE) CICS run-time library. The default is CEE.SCEECICS.

COBCOMP
The name of the COBOL compiler library. The default is IGY.SIGYCOMP.

COBLIB
The name of the LE link-edit library. The default is CEE.SCEELKED.

DFHLOAD
The name of the CICS load library. The default is DFH.SDFHLOAD.

DSNEXIT
The name of the DB2 DSNEXIT library. The default is DSN.DSNEXIT.

DSNLOAD
The name of the DB2 DSNLOAD library. The default is DSN.DSNLOAD.

ELA The name of the Enterprise Developer Server run-time high-level qualifier.

Note: Build scripts do not contain a job card, so do not add one.

Customizing the EGL Java Runtime

Enterprise Developer Options for z/OS provides the jar files needed to compile and execute EGL program that have been generated as Java programs. The jar files are named fda.jar and fdaj.jar and after installation are in the /usr/lpp/EnterpriseDeveloper/EGLJava directory under USS.

In order to use these files for compilation or execution, they should be included in any appropriate CLASSPATH settings. For example, in a shell script or .profile you could include the following (on one line):

```
export CLASSPATH=$CLASSPATH:
/usr/lpp/EnterpriseDeveloper/EGLJava/fda.jar:
/usr/lpp/EnterpriseDeveloper/EGLJava/fdaj.jar
```

See “Customizing the USS Remote Build Server” on page 10 for how the supplied shell script ccubldw.sh sets the variables.

Chapter 5. Activating the IBM WebSphere Developer for zSeries Common Access Repository Manager

Refer to *Program Directory for IBM WebSphere Developer for zSeries Common Access Repository Manager* (GI10-3367) for the SMP/E installation instructions for CARMA.

After installing CARMA, you must configure CARMA by following these steps:

1. Configure the CARMA server on your MVS host
2. **Optional:** Configure the sample RAMs
3. **Optional:** If you plan to use CARMA from WD/z, configure the CARMA components on your USS host
4. **Optional:** Restrict access on the initialization files and VSAM clusters. Under most circumstances, only System Administrators and CARMA developers will need to write to these files, while other users will only require read access.

Configuring the CARMA MVS components

Follow these steps to configure your MVS host:

1. Configure the *hlq*.SCRACLST(CRASUBMT) CLIST (where *hlq* is your high-level qualifier) to use your chosen high-level qualifier. Edit the following lines in your CLIST file to use your high-level qualifier instead of #hlq and your chosen library for your compiled RAMs instead of #ramlib:

```
//STEPLIB   DD DSN=#hlq.SCRALOAD,DISP=SHR
//          DD DSN=#ramlib,DISP=SHR
//CRADEF    DD DSN=#hlq.VSAMV.CRADEF,DISP=SHR
//CRAMSG    DD DSN=#hlq.VSAMV.CRAMSG,DISP=SHR
//CRASTRS   DD DSN=#hlq.VSAMV.CRASTRS,DISP=SHR
//CRARAM1   DD DSN=#hlq.VSAMV.CRARAM1,DISP=SHR
//CRARAM2   DD DSN=#hlq.VSAMV.CRARAM2,DISP=SHR
```

2. Configure the *hlq*.SCRASAM(CRAMREPR) JCL script to REPRO the CARMA VSAM message file (CRAMINIT). Edit the following lines in CRAMREPR to use your high-level qualifier instead of #hlq and your volume serial instead of #volume:

```
//DASD1    DD  VOL=SER=#volume,UNIT=SYSDA,DISP=SHR
        DEL #hlq.VSAMV.CRAMSG
//INPUT    DD  DSN=#hlq.SCRAVSAM(CRAMINIT),DISP=SHR
  DEF CLUSTER (NAME('#hlq.VSAMV.CRAMSG')       -
               KEYS(19 0) VOLUMES(#volume))     -
      REPRO INFILE(INPUT) ODS(#hlq.VSAMV.CRAMSG)
  LISTCAT ENT('#hlq.VSAMV.CRAMSG') ALL
```

   After making the modifications, submit this JCL. Upon successful submission, you should receive a return code of 8.

3. Configure the *hlq*.SCRASAM(CRAREPR) JCL script to REPRO the CARMA CAF VSAM definition file (CRAINIT). Edit the following lines in CRAREPR to use your high-level qualifier instead of #hlq and your volume serial instead of #volume:

```
//DASD1    DD  VOL=SER=#volume,UNIT=SYSDA,DISP=SHR
        DEL #hlq.VSAMV.CRADEF
//INPUT    DD  DSN=#hlq.SCRAVSAY(CRAINIT),DISP=SHR
  DEF CLUSTER (NAME('#hlq.VSAMV.CRADEF')       -
               KEYS(8 0) VOLUMES(#volume))     -
      REPRO INFILE(INPUT) ODS(#hlq.VSAMV.CRADEF)
  LISTCAT ENT('#hlq.VSAMV.CRADEF') ALL
```

   After making the modifications, submit this JCL. Upon successful submission, you should receive a return code of 8.

Note: The JCL script, CRAREPR, can be used to update the CRADEF VSAM cluster at a later time. To update the cluster, you must point the INPUT DD statement to your chosen sequential data set instead of CRAINIT. Refer to the *IBM WebSphere Developer for zSeries Version 6.0.1 Common Access Repository Manager Developer's Guide* (SC31-6914) for more information on defining this sequential data set.

4. Configure the *hlq*.SCRASAM(CRASREPR) JCL script (where *hlq* is your high-level qualifier) to use your high-level qualifier and volume serial. Edit the following lines in CRASREPR to use your high-level qualifier instead of #hlq and your volume serial instead of #volume:

```
//DASD1    DD  VOL=SER=#volume,UNIT=SYSALLDA,DISP=SHR
       DEL #hlq.VSAMV.CRASTRS
//INPUT    DD  DSN=#hlq.SCRAVSAY(CRASINIT),DISP=SHR
  DEF CLUSTER (NAME('#hlq.VSAMV.CRASTRS')       -
               KEYS(21 0) VOLUMES(#volume))     -
      REPRO INFILE(INPUT) ODS(#hlq.VSAMV.CRASTRS)
  LISTCAT ENT('#hlq.VSAMV.CRASTRS') ALL
```

After making the modifications, submit this JCL. If CRASTRS was created successfully, you will receive a return code of 8.

CAUTION:
Failure to provide the correct volume parameter will prevent successful completion of this JCL.

Note: The JCL script, CRASREPR, can be used to update the CRASTRS VSAM cluster at a later time. To update the cluster, you must point the INPUT DD statement to your chosen sequential data set instead of CRASINIT. Refer to the *IBM WebSphere Developer for zSeries Version 6.0.1 Common Access Repository Manager Developer's Guide* (SC31-6914) for more information on defining this sequential data set.

Configuring the sample RAMs

Follow the instructions in the sections below for the sample RAMs you want to configure.

Note: The sample RAMs are provided for the purpose of testing the configuration of your CARMA environment and as examples for developing your own RAMs. **Do NOT use the provided sample RAMs in a production environment.**

Configuring the SCLM RAM

Configure the *hlq*.SCRASAM(CRALREPR) JCL script (where *hlq* is your high-level qualifier) to REPRO the SCLM RAM's VSAM message file (CRALINIT). Edit the following lines in CRALREPR to use your high-level qualifier instead of #hlq and your volume serial instead of #volume:

```
//DASD1    DD  VOL=SER=#volume,UNIT=SYSALLDA,DISP=SHR
       DEL #hlq.VSAMV.CRARAM2
//INPUT    DD  DSN=#hlq.SCRAVSAM(CRALINIT),DISP=SHR
  DEF CLUSTER (NAME('#hlq.VSAMV.CRARAM2')       -
               KEYS(19 0) VOLUMES(#volume))     -
      REPRO INFILE(INPUT) ODS(#hlq.VSAMV.CRARAM2)
  LISTCAT ENT('#hlq.VSAMV.CRARAM2') ALL
```

After making the modifications, submit this JCL. Upon successful submission, you should receive a return code of 8.

Configuring the PDS RAM

Configure the *hlq*.SCRASAM(CRATREPR) JCL script (where *hlq* is your high-level qualifier) to REPRO the PDS RAM's VSAM message file (CRATINIT). Edit the following lines in CRATREPR to use your high-level qualifier instead of #hlq and your volume serial instead of #volume:

```
//DASD1    DD  VOL=SER=#volume,UNIT=SYSALLDA,DISP=SHR
       DEL #hlq.VSAMV.CRARAM1
//INPUT    DD  DSN=#hlq.SCRAVSAM(CRATINIT),DISP=SHR
  DEF CLUSTER (NAME('#hlq.VSAMV.CRARAM1')       -
               KEYS(19 0) VOLUMES(#volume))     -
      REPRO INFILE(INPUT) ODS(#hlq.VSAMV.CRARAM1)
  LISTCAT ENT('#hlq.VSAMV.CRARAM1') ALL
```

After making the modifications, submit this JCL. Upon successful submission, you should receive a return code of 8.

Compiling Skeleton RAM

You must compile Skeleton RAM before it can be used. First, modify the Skeleton RAM compilation JCL, *hlq*.SCRASAM(CRARAMCM) (where *hlq* is your high-level qualifier), by following these steps:

1. Replace #clientlib with *hlq*.SCRASAM(CRARAMSA).
2. Replace #target with *hlq*.SCRALOAD(TESTRAM).
3. Replace #header with *hlq*.SCRASAM.
4. Replace #sidedeck with the PDS you have chosen for storing your side decks (be sure to allocate this PDS first). If desired, #sidedeck may be replaced with 'DUMMY' so that no side deck is produced.

Note: In the above steps, *hlq* should be replaced by your high-level qualifier.

After making the modifications, submit this JCL. Upon successful submission, you should receive a return code of 4.

Configuring the CARMA USS components

Follow these steps to configure the USS CARMA components, which were installed with the WD/z RSE+ICU installation (FMID: H001600), on your USS host (where *H001600_INSTALLATION_DIR* is the USS directory where you installed the WD/z RSE+ICU components):

1. Edit the *H001600_INSTALLATION_DIR*/rse/lib/CRASRV.properties file as follows:
   a. Define the initial CRASRV port as the port that CRASRV will use to send information between CARMA components internal to the host. Communication on this port is confined to your host machine.
   b. Define the range as the range of ports on which to attempt to connect CRASRV.
   c. Define the location of the REXX submit script to CRASRV (this should default to *H001600_INSTALLATION_DIR*/rse/lib/rexxsub).

   Example: The following is an excerpt from an example CRASRV.properties file:

   ```
   port.start=2000
   port.range=200
   startup.script=/usr/lpp/wd4z/rse/lib/rexxsub
   ```

2. Configure the *H001600_INSTALLATION_DIR*/rse/lib/rexxsub CRASRV startup script to point to the location of your CRASRV start-up CLIST and fill in the appropriate value for DSNAME (*hlq*.SCRACLST(CRASUBMT), where *hlq* is your high-level qualifier).

   Example: The following is an excerpt from an example rexxsub script in which the high-level qualifier is hlq:

```
/* REXX */
say 'This was passed :' __argv.0 'arguments:'
/* Configure the CLIST */
DSNAME= "'hlq.scraclst(crasubmt)'"
say 'Running CLIST: ' DSNAME
```

Chapter 6. WebSphere Developer for zSeries Bidirectional support

WebSphere Developer for zSeries Service Flow Modeler (SFM) and XML Services for the Enterprise (XSE) components support different formats of Arabic and Hebrew interface messages, as well as correct bidirectional data presentation and editing in all editors and views. In terminal applications, both left-to-right and right-to-left screens are supported, as well as Numeric fields and fields with opposite-to-screen orientation.

Additional bidirectional features and functionality include the following:

- The SFM service requestor dynamically specifies bidirectional attributes of interface messages.
- Bidirectional data processing in SFM flows is based on bidirectional attributes (text type, text orientation, numeric swapping, and symmetric swapping.) These attributes can be specified in different stages of flow creation for both interface and terminal flows.
- SFM- and XSE-generated runtime code includes conversion of data between fields in messages that have different bidirectional attributes.

Place the programs FEJBDTRN and FEJBDTRE in the RPL concatenation and set autoinstall to autoactive.

If autoninstall is set to autoINactive, define the programs FEJBDTRN and FEJBDTRE to CICS using the CEDA command, for example:

```
CEDA DEF PROG(FEJBDTRN) LANG(LE) G(xxx)
```

Bidirectional runtime support

Service Flow Modeler and XML Services for the Enterprise bidirectional transformations are performed in the CICS Service Flow Runtime environment by the FEJBDTRN module. This module is supplied with the WebSphere Developer for zSeries host code. It can be copied into any library that is allocated as PDSE and appears in the CICS startup job DFHRPL list. The FEJBDTRN module is called when bidirectional conversions are needed in SFM- and XSE-generated code (when mapping is generated between fields in messages that have different bidirectional attributes.)

For Hebrew flows, you can also use the stand-alone FEJBDTRE version of the bidirectional conversion routine. FEJBDTRE can be copied into any library in the DFHRPL list (either PDS or PDSE). Functionally there is no difference between FEJBDTRN and FEJBDTRE. In order to generate CICS SFR runtime flow with calls to FEJBDTRE, you must change the default bidirectional module in the SFM bidirectional preferences.

Additionally, XSE-generated code can support BIDI transformation in environments other than CICS SFR (for example, batch applications). You can cause the XSE generators to include calls to the bidirectional conversion routines by specifying the appropriate BIDI transformation options in XSE generation wizards and linking the generated programs with the appropriate bidirectional conversion library. By

default, XSE generates calls to FEJBDTRN but you can change this default to FEJBDTRE by following the procedure outlined in the XSE section of Readme file.

For additional information, see Bidirectional considerations in the References section of Transforming CICS applications using Service Flow Modeler

Notices

Note to U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing IBM Corporation North Castle Drive Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR CONDITIONS OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
P.O. Box 12195, Dept. TL3B/B503/B313
3039 Cornwallis Rd.
Research Triangle Park, NC 27709-2195
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of 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.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples may include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

(C) (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. (C) Copyright IBM Corp. 2000, 2005. All rights reserved.

Trademarks and service marks

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

- CICS
- DB2
- IBM
- IMS™
- MVS
- OS/390
- RACF
- WebSphere
- z/OS

- zSeries

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Microsoft®, Windows, Windows NT®, and the Windows logo are trademarks or registered trademarks of Microsoft Corporation in the United States, or other countries, or both.

UNIX is a registered trademark of The Open Group.

Other company, product, and service names, which may be denoted by a double asterisk(**), may be trademarks or service marks of others.

Readers' Comments — We'd Like to Hear from You

IBM WebSphere Developer for zSeries Version 6.0.1
WebSphere Developer for zSeries
Host Configuration Guide

Publication No. SC31-6930-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name

Address

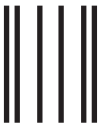
Company or Organization

Phone No.

Readers' Comments — We'd Like to Hear from You
SC31-6930-00

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Information Development
Department G7IA / Bldg. 503
P.O. Box 12195
Research Triangle Park, NC
27709-2195

Fold and Tape | **Please do not staple** | Fold and Tape

SC31-6930-00

Cut or Fold
Along Line

Program Number: 5724-L44

Printed in USA

SC31-6930-00