

IBM Rational Developer for System z



Host Configuration Guide

Version 7.1.1

IBM Rational Developer for System z



Host Configuration Guide

Version 7.1.1

Note

Before using this document, read the general information under "Notices" on page 123.

Second edition (December 2007)

This edition applies to IBM Rational Developer for System z Version 7.1.1 (program number 5724-T07) 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
USA

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

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

© Copyright International Business Machines Corporation 2005, 2007. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures	v
--------------------------	----------

Tables	vii
-------------------------	------------

About this book	ix
----------------------------------	-----------

Who should read this book	ix
Referenced publications	ix

Chapter 1. Installing and configuring the host components. 1

Pre-installation considerations	2
Pre-configuration considerations	2
Required configuration of requisite products and software	3
User ID considerations	4
Server considerations	4
Required permissions to implement the configuration tasks	5
Pre-deployment considerations	6
IBM Rational Developer for System z, FMID HHOP710	6
IBM Common Access Repository Manager (CARMA), FMID HCMA710	6

Chapter 2. Installation and configuration changes. 9

Changes between version 7.0 and version 7.1	9
IBM Rational Developer for System z, FMID HHOP710	9
IBM Common Access Repository Manager (CARMA), FMID HCMA710	9
Changes between version 6.0.1 and version 7.0	10
IBM WebSphere Developer for System z, FMID HHOP700	10
IBM Common Access Repository Manager (CARMA), FMID HCMA700	11
Backing up previously configured files	11

Chapter 3. Activating Developer for System z MVS components. 15

Set MAXASSIZE in SYS1.PARMLIB(BPXPRMxx)	15
APF authorize hlq.SFEKLOAD	15
Customize FEJJCNF, the JES Job Monitor configuration file	16
Customize the JES Job Monitor startup JCL.	18
JES Job Monitor tracing	18
Run JES Job Monitor as an STC.	18
Server permissions	20
Verification of JES Job Monitor startup JCL.	20
JES spool access and security	20
Conditional spool access	20
Available commands	21
Limiting access to spool files	21
Customize ELAXF*, remote build procedures	21

(Optional) Define an APPC transaction for the TSO Commands service	23
Preparation	24
Implementation	25
(Optional) Customize ELAXM*, DB2 stored procedure members	26
(Optional) Customize CICS Bidirectional language support (bidi).	27
(Optional) Customize Application Deployment Manager (ADM).	28
CRD repository	29
CICS primary connection region	29
Pipeline Message Handler	30
(Optional) CICS non-primary connection regions	30

Chapter 4. Activating Developer for System z z/OS UNIX components . . . 31

Saving the rsed.envvars configuration file in another directory	32
Customize rsed.envvars, the configuration file for RSE	33
(Optional) Defining the PORTRANGE available for RSE.	38
(Optional) Defining extra Java startup parameters with _RSE_*OPTS	38
INETD daemon and RSE REXEC/SSH setup	40
INETD RSE daemon set up	40
INETD REXEC (or SSH) set up	42
Customize ISPF.conf, ISPF configuration file	43
Verify RSE server set up	43
Port availability	44
REXEC connection	45
REXEC/SSH shell script	46
RSE daemon connection	46
JES Job Monitor connection	47
TSO Commands service connection (using SCLMDT)	47
TSO Commands service connection (using APPC)	48
(Optional) Customize ssl.properties, RSE SSL configuration	49
(Optional) Customize rsecomm.properties, RSE trace configuration	49
(Optional) Customize projectcfg.properties, host projects configuration	50
(Optional) Customize FMIEXT.properties, File Manager integration	51

Chapter 5. (Optional) Activating IBM Common Access Repository Manager (CARMA). 53

Customizing the CARMA MVS components	53
Customizing the CARMA z/OS UNIX components	54
(Optional) Activating the sample Repository Access Managers (RAMs)	56
Activating the SCLM RAM	56

Activating the PDS RAM	57
Activating the skeleton RAM	57

Chapter 6. (Optional) Activating IBM Software Configuration and Library Manager (SCLM) Developer Toolkit . . . 59

Chapter 7. Developer for System z client considerations 61

Chapter 8. Performance considerations 63

Use zFS file systems	63
Avoid use of STEPLIB	63
Improve access to system libraries	64
Language Environment (LE) runtime libraries	64
Application development	64
Improving performance of security checking	65
Class sharing between JVMs	65
Enable class sharing	65
Cache size limits	65
Cache security	66
SYS1.PARMLIB(BPXPRMxx)	66
Disk space	66
Cache management utilities	66
Fixed Java heap size	67
Workload management	68

Appendix A. Running multiple instances of Developer for System z . . 69

Identical software level, different configuration files	69
All other situations	69

Appendix B. Troubleshooting configuration problems 73

Location of log files	73
JES Job Monitor logging	74
APPC transaction (TSO Commands service) logging	74
RSE logging	74
fekfivpc IVP test logging	75
Fault Analyzer Integration logging	75
File Manager Integration logging	75
CARMA logging	75
Dump files	76
MVS dumps	76
Java dumps	76
z/OS UNIX dump locations	77
Program Control authorization for RSE programs	77
Reserved TCP/IP ports	78
Address Space size	79
INETD requirements	80
Limitations set in SYS1.PARMLIB(BPXPRMxx)	80
Limitations stored in the security profile	80
Limitations enforced by system exits	80
Error feedback tracing	80
APPC transaction and TSO Commands service	81
Miscellaneous information	83

System limits	83
Known Issues	83
Host Connect Emulator	84
Contacting IBM support	84

Appendix C. Setting up TCP/IP 87

Hostname dependency	87
Understanding resolvers	87
Understanding search orders of configuration information	88
Search orders used in the z/OS UNIX environment	88
Base resolver configuration files:	88
Translate tables:	89
Local host tables:	89
Applying this to Developer for System z	90

Appendix D. Setting up INETD 93

inetd.conf	93
ETC.SERVICES	94
Search order used in the z/OS UNIX environment	95
Search order used in the native MVS environment	95
PROFILE.TCPIP port definitions	95
/etc/inetd.pid	96
Startup	96
/etc/rc	97
/etc/inittab	97
BPXBATCH	97
Shell session	98
Security	98
Developer for System z requirements	99
INETD	99
RSE daemon	99

Appendix E. Setting up SSL 101

Clone the existing RSE setup	102
Determine which key file(s) to use	102
Create a key store with keytool	103
Create a key database (daemon only)	104
Create a key ring with RACF	104
Create a key database with gskkyman	105
Activate SSL by updating ssl.properties	109
Test the connection	109

Appendix F. Setting up APPC 113

VSAM	113
VTAM	114
SYS1.PARMLIB(APPCPMxx)	115
SYS1.PARMLIB(ASCHPMxx)	116
Activating APPC changes	117
Defining the TSO Commands service transaction	117

Glossary 119

Notices 123

Trademarks and service marks	125
--	-----

Figures

1.	FEJCNFG, JES Job Monitor configuration file	16
2.	JES Job Monitor JCL	19
3.	REXX for APPC ISPF panels	24
4.	rsed.envvars – RSE configuration file	34
5.	ISPF.conf - ISPF configuration file	43
6.	ssl.properties – SSL configuration file	49
7.	rsecomm.properties – Logging configuration file	50
8.	projectcfg.properties – Host based projects configuration file.	50
9.	FMIEXT.properties – File Manager configuration file.	51
10.	CRASRV.properties – CARMA configuration file	55
11.	INETD startup JCL	97
12.	Import Host Certificate	110
13.	Preferences	111
14.	JCL to create APPC VSAMs	114
15.	SYS1.SAMPLIB(ATBAPPL)	115
16.	SYS1.PARMLIB(APPCPMxx)	115
17.	SYS1.PARMLIB(ASCHPMxx)	117

Tables

1. Referenced publications	ix	7. Sample ELAXF* procedures	22
2. Developer for System z installation and configuration matrix	1	8. ELAXF* high level qualifier checklist	22
3. Customized MVS members overview	11	9. APPC transaction checklist	24
4. Customized z/OS UNIX files overview	13	10. Sample ELAXM* DB2 stored procedure members	26
5. Customization in non Developer for System z libraries.	14	11. Optional configuration files	32
6. JES Job Monitor console commands	21	12. Developer for System z client checklist	61
		13. Local definitions available to resolver	92

About this book

This book discusses the configuration of the IBM Rational Developer for System z functions. It includes instructions on how to configure IBM Rational Developer for System z servers on your z/OS® host system.

From here on, the following names are used in this manual:

- *IBM Rational Developer for System z* is called *Developer for System z*
- *IBM Rational Developer for System z Common Access Repository Manager* is called *Common Access Repository Manager*, sometimes abbreviated to *CARMA*
- *IBM® Software Configuration and Library Manager (SCLM) Developer Toolkit* is called *SCLM Developer Toolkit*, sometimes abbreviated to *SCLMDT*

Note: The configuration information found in this document is for IBM Rational Developer for System z Version 7.1.1.

For earlier releases, including IBM WebSphere Developer for System z, IBM WebSphere Developer for zSeries and WebSphere Studio Enterprise Developer, use the configuration information found in the Host Configuration Guide and Program Directories for those releases.

Who should read this book

This book is intended for system programmers installing and configuring IBM Rational Developer for System z Version 7.1.1 on their z/OS host system. To use this book, you need to be familiar with the z/OS UNIX® and MVS™ host systems.

Referenced publications

The following publications are referenced in this document:

Table 1. Referenced publications

Publication title	Order number	Reference	Reference Link
Java Diagnostic Guide	SC34-6650	Java 5.0	http://www.ibm.com/developerworks/java/jdk/diagnosis/index.html
Java SDK and Runtime Environment User Guide		Java 5.0	http://www-03.ibm.com/servers/eserver/zseries/software/java/
Program Directory for IBM Rational Developer for System z	GI11-8298-00	RD/z	http://www-306.ibm.com/software/awdtools/rdz/library/index.html
Program Directory for IBM Rational Developer for System z Common Access Repository Manager	GI11-8299-00	RD/z	http://www-306.ibm.com/software/awdtools/rdz/library/index.html

Table 1. Referenced publications (continued)

Publication title	Order number	Reference	Reference Link
Program Directory for IBM Software Configuration and Library Manager (SCLM) Developer Toolkit	GI11-8306-00	RD/z	http://www-306.ibm.com/software/awdtools/rdz/library/index.html
Rational Developer for System z Application Deployment Manager User's Guide	SC23-7661	RD/z	http://www-306.ibm.com/software/awdtools/rdz/library/index.html
Rational Developer for System z Common Access Repository Manager Developer's Guide	SC23-7660	RD/z	http://www-306.ibm.com/software/awdtools/rdz/library/index.html
Rational Developer for System z Host Configuration Guide	SC23-7658	RD/z	http://www-306.ibm.com/software/awdtools/rdz/library/index.html
Rational Developer for System z Host Planning Guide	GI11-8296-00	RD/z	http://www-306.ibm.com/software/awdtools/rdz/library/index.html
SCLM Developer Toolkit Installation and Customization Guide	SC23-8504	RD/z	http://www-306.ibm.com/software/awdtools/rdz/library/index.html
ABCs of z/OS System Programming Volume 9	SG24-6989	Redbook	http://www.redbooks.ibm.com/
APPC and WebSphere Developer for System z	SC23-5885	Whitepaper	http://www-306.ibm.com/software/awdtools/rdz/library/index.html
Host Based Projects in WebSphere Developer for System z version 7.0		Whitepaper	http://www-306.ibm.com/software/awdtools/rdz/library/index.html
Setting up SSL for RSE Communication	SC23-5816	Whitepaper	http://www-306.ibm.com/software/awdtools/rdz/library/index.html
Communications Server bookshelf	F1A1BK61	z/OS 1.7	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Communications Server IP Configuration Guide	SC31-8775	z/OS 1.7	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Communications Server IP Configuration Reference	SC31-8776	z/OS 1.7	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Communications Server IP Diagnosis Guide	GC31-8782	z/OS 1.7	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/

Table 1. Referenced publications (continued)

Publication title	Order number	Reference	Reference Link
Communications Server IP System Administrator's Commands	SC31-8781	z/OS 1.7	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Communications Server SNA Network Implementation Guide	SC31-8777	z/OS 1.7	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Cryptographic Services System SSL Programming	SC24-5901	z/OS 1.7	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Language Environment Customization	SA22-7564	z/OS 1.7	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Language Environment Debugging Guide	GA22-7560	z/OS 1.7	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
MVS bookshelf	IEA2BK60	z/OS 1.7	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
MVS Initialization and Tuning Guide	SA22-7591	z/OS 1.7	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
MVS Initialization and Tuning Reference	SA22-7592	z/OS 1.7	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
MVS JCL Reference	SA22-7597	z/OS 1.7	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
MVS Planning APPC/MVS Management	SA22-7599	z/OS 1.7	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
MVS Planning Workload Management	SA22-7602	z/OS 1.7	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
MVS System Commands	SA22-7627	z/OS 1.7	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Security Server RACF Command Language Reference	SA22-7687	z/OS 1.7	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Security Server RACF Security Administrator's Guide	SA22-7683	z/OS 1.7	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
UNIX System Services Command Reference	SA22-7802	z/OS 1.7	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
UNIX System Services File System Interface Reference	SA22-7808	z/OS 1.7	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
UNIX System Services Planning	GA22-7800	z/OS 1.7	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
UNIX System Services User's Guide	SA22-7801	z/OS 1.7	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/

Chapter 1. Installing and configuring 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 2. Developer for System z installation and configuration matrix

If you require this IBM Rational Developer for System z function	You must install these FMIDs	And you will find installation and configuration information here
<ul style="list-style-type: none">• Host Connectivity• JES Connectivity• Remote Compile• Error Feedback for Remote Compile• Remote Debugging• DB2[®] Stored Procedures• IMS[™] MFS Screen Support• CICS[®] BMS Map Support• CICS Bidirectional language support (bidi)• Application Deployment Manager (ADM)• File Manager integration• Fault Analyzer integration	HHOP710, HSD3310*	<ul style="list-style-type: none">• <i>Program Directory for IBM Rational Developer for System z (GI11-8298-00)</i>• <i>Rational Developer for System z Host Configuration Guide (SC23-7658)</i>• <i>Rational Developer for System z Host Planning Guide (GI11-8296-00)</i> optional <ul style="list-style-type: none">• <i>Program Directory for IBM Software Configuration and Library Manager (SCLM) Developer Toolkit (GI11-8306-00)</i>• <i>SCLM Developer Toolkit Installation and Customization Guide (SC23-8504)</i>
<ul style="list-style-type: none">• Common Software Configuration Management Access (CARMA)	HCMA710, HHOP710**	<ul style="list-style-type: none">• <i>Program Directory for IBM Rational Developer for System z Common Access Repository Manager (GI11-8299-00)</i> optional <ul style="list-style-type: none">• <i>Program Directory for IBM Rational Developer for System z (GI11-8299-00)</i>• <i>Rational Developer for System z Host Configuration Guide (SC23-7658)</i>• <i>Rational Developer for System z Host Planning Guide (GI11-8296-00)</i>
<ul style="list-style-type: none">• Software Configuration and Library Manager (SCLM) Developer Toolkit	HSD3310	<ul style="list-style-type: none">• <i>Program Directory for IBM Software Configuration and Library Manager (SCLM) Developer Toolkit (GI11-8306-00)</i>• <i>SCLM Developer Toolkit Installation and Customization Guide (SC23-8504)</i>

(*) Developer for System z requires a connection to the TSO Commands service on z/OS. This connection is provided by one of the following:

1. Installing and configuring SCLM Developer Toolkit, FMID HSD3310 (the default)
2. Installing and configuring an APPC transaction

(**) CARMA requires a host based interface. HHOP710 provides this function, or you may use a custom built, host based, interface and omit installing HHOP710

Pre-installation considerations

- If you are a previous user of IBM Rational Developer for System z, IBM WebSphere Developer for System z, IBM WebSphere Developer for zSeries or WebSphere Studio Enterprise Developer, it is recommended that you save the related customized files before installing the upgrade to IBM Rational Developer for System z Version 7.1.1. Refer to “Backing up previously configured files” on page 11 for an overview of files that required customization.
- Read Appendix A, “Running multiple instances of Developer for System z,” on page 69 if you plan on running multiple instances of Developer for System z.
- Since version 7.1, Developer for System z provides two access methods for the TSO Commands service, which executes TSO/ISPF commands, either implicitly by the product, or explicitly by the user.

- Calling a SCLM Developer Toolkit function. This method requires the installation and configuration of SCLMDT, FMID HSD3310, which is shipped with Developer for System z. This method is the default used in the sample members.

The following SCLMDT customization steps must be performed, which are described in the *SCLM Developer Toolkit Installation and Customization Guide* (SC23-8504):

- Create the directory structure and configuration files
- Customize ISPF.conf

Note: Neither the SCLM product, nor the Eclipse based client are required to support this function of SCLMDT.

- An APPC transaction (as in pre version 7.1 releases). This method requires the setup and configuration of APPC, which is part of z/OS.
- The bidi load module changed in version 7.0 (name and coding), requiring the removal of the older modules and configuration of the new one.
- Bidi code that has been created with previous releases (pre version 7.0) must be RECOMPILED in order to use the new FEJBDTRX module.

For detailed instructions on the SMP/E installation for each of the FMIDs, refer to the appropriate program directory, as listed in Table 2 on page 1.

Pre-configuration considerations

Developer for System z has a list of prerequisite software that must be installed and operational before the product will work. There is also a list of corequisite software to support specific features of Developer for System z. These requisites must be installed and operational at runtime for the corresponding feature to work as designed.

Refer to *Rational Developer for System z Host Planning Guide* (GI11-8296-00) to get a list of prerequisites and corequisites for your version of Developer for System z.

Attention: 64-bit Java™ is NOT supported.

Required configuration of requisite products and software

Consult your MVS system programmer, security administrator and TCP/IP administrator to check if the requisite products and software are installed, tested, and working.

- Since version 7.1, Developer for System z provides two methods for executing TSO/ISPF commands, either implicitly by the product, or explicitly by the user.
 - Calling a SCLM Developer Toolkit function. This method requires the installation and configuration of SCLMDT, FMID HSD3310, which is shipped with Developer for System z. This method is the default used in the sample members.

The following SCLMDT customization steps must be performed, which are described in the *SCLM Developer Toolkit Installation and Customization Guide* (SC23-8504):

- Create the directory structure and configuration files
- Customize ISPF.conf

Note: Neither the SCLM product, nor the Eclipse based client are required to support this function of SCLMDT .

- An APPC transaction (as in pre version 7.1 releases). This method requires the installation and configuration of APPC, which is part of z/OS.
- The C/C++ DLL class library CBC.SCLBDLL and the Language Environment® (LE) runtime libraries CEE.SCEERUN and CEE.SCEERUN2 must be in LINKLIST.
- All z/OS UNIX users must have READ and EXECUTE access to the Java directories.
- Developer for System z is dependant upon INETD for setting up the client-host connection. Refer to Appendix D, "Setting up INETD," on page 93 for more information on REXEC and SSH.
- Remote (host based) actions for z/OS UNIX subprojects require that REXEC or SSH is active on the host.
- Developer for System z is dependant upon TCP/IP having the correct hostname when it is initialized. This implies that the different TCP/IP and Resolver configuration files must be set up correctly. For TCP/IP and Resolver customization information, see Appendix C, "Setting up TCP/IP," on page 87 and "TCPIP.DATA configuration statements" in the *Communications Server IP Configuration Reference* (SC31-8776).

You can test your TCP/IP configuration with the TSO command **HOMETEST**. Refer to "TSO Commands" in the *Communications Server IP System Administrator's Commands* (SC31-8781) for more information on this command.

Example output of the **HOMETEST** command:

Running IBM MVS TCP/IP CS V1R7 TCP/IP Configuration Tester

The FTP configuration parameter file used will be "SYS1.TCPPARMS(FTPDATA)"

TCP Host Name is: CDFMVS08

Using Name Server to Resolve CDFMVS08

The following IP addresses correspond to TCP Host Name: CDFMVS08
9.42.112.75

The following IP addresses are the HOME IP addresses defined in PROFILE.TCPIP:
9.42.112.75
127.0.0.1

All IP addresses for CDFMVS08 are in the HOME list!

Hometest was successful - all Tests Passed!

User ID considerations

The user ID of a Developer for System z user must have (at least) the following attributes:

- TSO access (with a normal region size).
- An OMVS segment defined to the security system (e.g. RACF®), both for the user ID and its default group. This OMVS segment can be either an individual or a system wide default one. See *Security Server RACF Security Administrator's Guide (SA22-7683)* for more information on the default OMVS segment.
 - A home directory (with WRITE, READ and EXECUTE access) allocated for the user and identified in the OMVS segment.
 - The PROGRAM field in the OMVS segment should be /bin/sh or other valid z/OS UNIX shell such as /bin/tcsh.
 - The ASSIZEMAX field should not be set, so that system defaults will be used.
 - The user ID does not require UID 0.

Example (command **LISTUSER userid NORACF OMVS**):

USER=userid

```
OMVS INFORMATION
-----
UID= 0000003200
HOME= /u/userid
PROGRAM= /bin/sh
CPUTIMEMAX= NONE
ASSIZEMAX= NONE
FILEPROCMA= NONE
PROCUSERMA= NONE
THREADSMA= NONE
MMAPAREAMA= NONE
```

- The user ID's default group requires a GID.

Example (command **LISTGRP group NORACF OMVS**):

GROUP group

```
OMVS INFORMATION
-----
GID= 0000003243
```

- READ and EXECUTE access to the Developer for System z installation and configuration directories, default /usr/lpp/wd4z/*.

Server considerations

The following host services, and thus their ports, must be available for the client to connect to. All other ports used by Developer for System z have host-only traffic. This means that, for Developer for System z, only the listed ports must be defined to your firewall protecting the host.

- INETD service to start the RSE server. Developer for System z supports multiple ways to do this:
 - RSE daemon, default port 4035
 - REXEC, default port 512
 - SSH, default port 22

- RSE server for client-host communication. Uses any available port by default, but can be limited to a specified range. Communication on this port can be encrypted using SSL.
- TN3270 Telnet service for the Host Connect Emulator, default port 23. Communication on this port can be encrypted using SSL (default port 992). The default port assigned to the TN3270 Telnet service depends on whether or not the user chooses to use encryption.

Note: Previous clients (version 7.0 and older) communicate directly with JES Job Monitor, default port 6715.

Note: During a remote debug session for Cobol, PL/I or Assembler, IBM Debug Tool for z/OS is invoked. This product communicates directly with the client. This communication is initiated on the host, and connects to port 8001 on the client.

INETD is a z/OS UNIX server process that is required for setting up Developer for System z client-host connections. INETD's environmental settings, which are passed on when starting a process, and the permissions for INETD's user ID must be set properly in order for INETD to start the RSE server.

- If INETD is started by JCL using BPXBATCH, the region size must be 0.
- If INETD is started from a TSO/OMVS session, the TSO region size must be 2096128 or larger.
- READ and EXECUTE access to the Developer for System z installation directories, default /usr/lpp/wd4z/*.
- BPX.DAEMON and possibly UID 0 permission.

See Appendix D, "Setting up INETD," on page 93 for more information on INETD permissions.

Remote Systems Explorer (RSE) is the Developer for System z component that provides core services like connecting the client to the host.

- Each client-host connection has a private RSE server.
- Besides the ports documented in this book for communicating with other Developer for System z servers, RSE server requires multiple ports for internal use.

Required permissions to implement the configuration tasks

The configuration of Developer for System z requires more than the typical system programmer permissions, so minimal assistance from others is to be expected. The following list highlights the most important areas:

- Update permission for system data sets (like SYS1.PARMLIB)
- Permission to issue console commands
- Permission to create z/OS UNIX directories
- Permission to update z/OS UNIX /etc directory
- Permission to set the z/OS UNIX program control bit
- Permission to define new TCP/IP ports
- BPX.DAEMON permission and possibly UID 0 to (re)start INETD
- WLM/RSM management permissions
- APPC management permissions to create a transaction program
- Security administrator permissions to query and create security profiles

- (optional) DB2 administrator permissions to create a stored procedure
- (optional) CICS administrator permissions to define programs to CICS

Pre-deployment considerations

Developer for System z and Common Access Repository Manager (CARMA) support cloning an installation to a different system, avoiding the need for a SMP/E install on each system.

The following data sets, directories and files are mandatory for deployment to other systems. If you copied a file to a different location for customization, then this file must replace its counterpart in the lists below.

Note: The lists below do not cover the deployment needs of the pre and co requisite software.

IBM Rational Developer for System z, FMID HHOP710

- hlq.SFEKLOAD(*)
- hlq.SFEKPROC(*)
- hlq.SFEKSAMP(FEJJCENFG)
- hlq.SFEKSAMP(FEJJJCL)
- hlq.SFEKSAMP(ELAXF*)
- /usr/lpp/wd4z/*
- optional parts
 - hlq.SFEKSAMP(FEJTSO) - TSO submit
 - hlq.SFEKSAMP(ELAXM*) - DB2 stored procedure
 - CICS definitions created by hlq.SFEKSAMP(ADNPCCSD) - ADM primary connection
 - CICS definitions created by hlq.SFEKSAMP(ADNARCSD) - ADM non-primary connection
 - VSAM created by hlq.SFEKSAMP(ADNVSAM) - ADM CRD server
 - load module created by hlq.SFEKSAMP(ADNCMSGH) - ADM Pipeline Handler
 - APPC transaction created by hlq.SFEKSAMP(FEKAPPCC) - APPC based TSO Commands service
 - /etc/SCLMDT/CONFIG/ISPF.conf - add the TSO Commands Server to the ISPF environment of SCLM Developer Toolkit

Note: hlq and /usr/lpp/wd4z are the high level qualifier (default FEK) and path used during the installation of the product.

IBM Common Access Repository Manager (CARMA), FMID HCMA710

- hlq.SCRALOAD(*)
- hlq.SCRACLIST(CRASUBMT)
- VSAMs created by the following hlq.SCRASAMP members
 - hlq.SCRASAMP(CRA\$VMSG) message VSAM
 - hlq.SCRASAMP(CRA\$VDEF) configuration VSAM
 - hlq.SCRASAMP(CRA\$SVSTR) custom information VSAM

Note: hlq is the high level qualifier (default CRA) used during the installation of the product.

Chapter 2. Installation and configuration changes

This section highlights installation and configuration changes compared to previous releases of the product. Refer to the related sections in this manual for more information.

Changes between version 7.0 and version 7.1

IBM Rational Developer for System z, FMID HHOP710

- **Added:** Setup choice - TSO/ISPF commands via an APPC transaction or via SCLM Developer Toolkit
- **Changed:** The APPC transaction exploits a new ISPF feature
- **Added:** The following customizable members are new
 - samplib ELAXFADT
 - samplib ADNCMSGH
 - /usr/lpp/wd4z/rse/lib/FMIEXT.properties
- **Changed:** The following members have moved
 - SFEKDLL(FEJBDTRX) -> SFEKLOAD(FEJBDTRX)
- **Changed:** The following customizable members have changed
 - samplib FEKFAPPCC
 - /usr/lpp/wd4z/rse/lib/rsed.envvars
 - /usr/lpp/wd4z/rse/lib/setup.env.zseries
 - /usr/lpp/wd4z/rse/lib/server.zseries

IBM Common Access Repository Manager (CARMA), FMID HCMA710

- **Changed:** Logging is written to CARMALOG DD statement
- **Changed:** The CARMA message VSAM (CRAMSG) and configuration VSAM (CRADEF) are updated
- **Added:** The following customizable members are new
 - samplib CRA#ECOB
 - samplib CRA#EPDS
 - samplib CRA#ERAM
 - samplib CRA#ESLM
- **Renamed:** The following customizable members are renamed
 - samplib CRAREPR -> CRA\$VDEF
 - samplib CRAMREPR -> CRA\$VMMSG
 - samplib CRASREPR -> CRA\$VSTR
 - samplib CRASALX -> CRA#ASLM
 - samplib CRACOBJ1 -> CRA#CCB1
 - samplib CRACOBJ2 -> CRA#CCB2
 - samplib CRACLICM -> CRA#CCLT
 - samplib CRARAMCS -> CRA#CPDS
 - samplib CRARAMCM -> CRA#CRAM

- samplib CRATREPR -> CRA#VPDS
- samplib CRALREPR -> CRA#VSLM
- samplib CRACLIRN -> CRA#XCLT
- **Changed:** The following customizable members have changed
 - clist CRASUBMT

Changes between version 6.0.1 and version 7.0

IBM WebSphere Developer for System z, FMID HHOP700

- **Changed:** IBM WebSphere Developer for System z, FMID HHOP700, now combines the functions of 5 separate products into 1 to reduce the installation work
 - v601 FMID H001600, IBM WebSphere Developer for zSeries RSE + ICU
 - v601 FMID H002600, IBM WebSphere Developer for zSeries JES Job Monitor
 - v601 FMID HEDS500, IBM WebSphere Developer for zSeries Options for z/OS (not all functions are included in HHOP700)
 - v601 FMID HBDI601, IBM WebSphere Developer for zSeries bidi support
 - (new) version 700, IBM Application Deployment Manager
- **Changed:** Installation jobs are renamed so that the steps to be performed are in alphabetical order
- **Changed:** SMP/E install now sets program control bits in z/OS UNIX
- **Changed:** REXX server coding is moved out of samplib to a new data set, SFEKPROC
- **Changed:** Bidi uses a new load module, all references to the old ones must be removed
- **Changed:** The JESNAME parameter is no longer used in the JES Job Monitor configuration file (samplib FEJJCNFG)
- **Changed:** The following members no longer need customization
 - /usr/lpp/wd4z/rse/lib/setup.env.zseries
- **Added:** The following members are new
 - samplib ELAXFCPC
 - samplib ELAXFCPP
 - samplib ELAXFMFS
 - /usr/lpp/wd4z/rse/lib/projectcfg.properties
- **Renamed:** The following customizable members are renamed
 - samplib FEKFRDIS -> FEKAPPCL
 - samplib FEKFRDEL -> FEKAPPCX
 - samplib FEKFRTAD -> FEKAPPCC
- **Removed:** The following members are no longer shipped
 - samplib ELAXFLNK
 - samplib FEJIVJCL
 - loadlib FEJBDTRN
 - loadlib FEJBDTRE

IBM Common Access Repository Manager (CARMA), FMID HCMA700

- **Changed:** Installation jobs are renamed so that the steps to be performed are in alphabetical order
- **Changed:** The following members no longer need customization
 - /usr/lpp/wd4z/rse/lib/rsed.envvars
 - /usr/lpp/wd4z/rse/lib/rexxsub

Backing up previously configured files

Before installing Developer for System z version 7.1.1, if you are a previous user of Developer for System z, it is recommended that you save the related customized files. Read Appendix A, “Running multiple instances of Developer for System z,” on page 69 before starting the customization if you plan on running multiple instances of Developer for System z.

Table 3 and Table 4 on page 13 give an overview of files that may have been customized for Developer for System z and CARMA version 6.0.1 and higher. Table 5 on page 14 lists customizations to data sets, prerequisite and corequisite products and software that occur during a Developer for System z and CARMA version 6.0.1 (and higher) installation.

Table 3. Customized MVS members overview

Member	Default location v6.0.1	Default location v7.0/v7.1	Purpose
FEJCNFG	hlq.SFEJSAMP (hlq = FEJ)	hlq.SFEKSAMP (hlq = FEK)	Configuration file for JES Job Monitor
FEJJJCL	hlq.SFEJSAMP (hlq = FEJ)	hlq.SFEKSAMP (hlq = FEK)	JCL for JES Job Monitor
FEJTSO	hlq.SFEJSAMP (hlq = FEJ)	hlq.SFEKSAMP (hlq = FEK)	JCL for TSO submits
FEKAPPCC	doesn't exist	hlq.SFEKSAMP (hlq = FEK)	JCL to create an APPC transaction
FEKAPPCL	doesn't exist	hlq.SFEKSAMP (hlq = FEK)	JCL to display an APPC transaction
FEKAPPCX	doesn't exist	hlq.SFEKSAMP (hlq = FEK)	JCL to delete an APPC transaction
FEKFRPAD	hlq.SFEKSAMP (hlq = FEK)	(new member name FEKAPPCC)	JCL to create an APPC transaction
FEKFRDIS	hlq.SFEKSAMP (hlq = FEK)	(new member name FEKAPPCL)	JCL to display an APPC transaction
FEKFRDEL	hlq.SFEKSAMP (hlq = FEK)	(new member name FEKAPPLX)	JCL to delete an APPC transaction

Table 3. Customized MVS members overview (continued)

Member	Default location v6.0.1	Default location v7.0/v7.1	Purpose
ELAXF*	hlq.SCCUSAMP (hlq = CCU)	hlq.SFEKSAMP (hlq = FEK)	JCL for remote project builds, etc.
ELAXMSAM	hlq.SCCUSAMP (hlq = CCU)	hlq.SFEKSAMP (hlq = FEK)	JCL procedure of the WLM address space for the PL/I and COBOL Stored Procedure Builder
ELAXMJCL	hlq.SCCUSAMP (hlq = CCU)	hlq.SFEKSAMP (hlq = FEK)	JCL for defining the PL/I and COBOL Stored Procedure Builder to DB2
ADNVSAM	doesn't exist	hlq.SFEKSAMP (hlq = FEK)	JCL for defining the ADM CRD repository
ADNPCCSD	doesn't exist	hlq.SFEKSAMP (hlq = FEK)	JCL for activating the CRD server in the primary CICS region
ADNCMSGH	doesn't exist	hlq.SFEKSAMP (hlq = FEK) (doesn't exist in version 7.0)	JCL for compiling the Pipeline Message Handler
ADNSMSGH	doesn't exist	hlq.SFEKSAMP (hlq = FEK)	Sample source code for the Pipeline Message Handler
ADNARCSD	doesn't exist	hlq.SFEKSAMP (hlq = FEK)	JCL for activating the CRD server in non-primary CICS regions
CRASUBMT	hlq.SCRACLST (hlq = CRA)	hlq.SCRACLST (help = CRA)	CLIST to submit JCL for a CARMA server
CRAMREPR	hlq.SCRASAM (hlq = CRA)	hlq.SCRASAMP (hlq = CRA) (new member name in version 7.1 CRA\$VMSG)	JCL to create the CARMA message VSAM

Table 3. Customized MVS members overview (continued)

Member	Default location v6.0.1	Default location v7.0/v7.1	Purpose
CRAREPR	hlq.SCRASAM (hlq = CRA)	hlq.SCRASAMP (hlq = CRA) (new member name in version 7.1 CRA\$VDEF)	JCL to create the CARMA configuration VSAM
CRASREPR	hlq.SCRASAM (hlq = CRA)	hlq.SCRASAMP (hlq = CRA) (new member name in version 7.1 CRA\$VSTR)	JCL to create the CARMA custom information VSAM
CRALREPR	hlq.SCRASAM (hlq = CRA)	hlq.SCRASAMP (hlq = CRA) (new member name in version 7.1 CRA#VSLM)	JCL to create the SCLM RAM's message VSAM
CRASALX	hlq.SCRASAM (hlq = CRA)	hlq.SCRASAMP (hlq = CRA) (new member name in version 7.1 CRA#ASLM)	JCL to create the SCLM RAM's data sets
CRATREPR	hlq.SCRASAM (hlq = CRA)	hlq.SCRASAMP (hlq = CRA) (new member name in version 7.1 CRA#VPDS)	JCL to create the PDS RAM's message VSAM

Note: hlq is the high level qualifier used during the installation of the product.
The IBM supplied defaults for hlq are listed, but may not apply to your site.

Table 4. Customized z/OS UNIX files overview

File	Default location v6.0.1	Default location version 7.0/version 7.1	Purpose
rsed.envvars	/usr/lpp/wd4z/rse/lib/	/usr/lpp/wd4z/rse/lib/	RSE configuration file
rsecomm.properties	/usr/lpp/wd4z/rse/lib/	/usr/lpp/wd4z/rse/lib/	RSE trace configuration file
ssl.properties	/usr/lpp/wd4z/rse/lib/	/usr/lpp/wd4z/rse/lib/	RSE SSL configuration file
setup.env.zseries	/usr/lpp/wd4z/rse/lib/	(no longer customized)	Export RSE environment variables
projectcfg.properties	(doesn't exist)	/usr/lpp/wd4z/rse/lib/	Host projects configuration file
FMIEXT.properties	(doesn't exist)	/usr/lpp/wd4z/rse/lib/ (doesn't exist in version 7.0)	File Manager configuration file
CRASRV.properties	/usr/lpp/wd4z/rse/lib/	/usr/lpp/wd4z/rse/lib/	CARMA configuration file
rexsub	/usr/lpp/wd4z/rse/lib/	(no longer customized)	CARMA z/OS UNIX REXX to execute MVS CRASUBMT CLIST

Note: /usr/lpp/rd4z is the path used during the installation of the products. The IBM supplied default is shown, but might not apply to your site.

Table 5. Customization in non Developer for System z libraries

Member/ File	Default location	Required customization
BPXPRMxx	SYS1.PARMLIB	Set MAXASSIZE to 2147483647 or larger
PROGxx	SYS1.PARMLIB	APF authorize hlq.SFEKLOAD
ASCHPMxx	SYS1.PARMLIB	Define an APPC transaction class for the TSO Commands service
services	/etc/	Define RSE daemon
inetd.conf	/etc/	Define RSE daemon
ISPF.conf	/etc/SCLMDT/ CONFIG/	Define location of the TSO Commands Server
/	APPC	Define an APPC transaction for the TSO Commands service
/	WLM	Associate APPC transaction program with a TSO performance group
/	WLM	Assign an application environment for the DB2 stored procedure

Chapter 3. Activating Developer for System z MVS components

Before installing the 7.1.1 version, if you are a previous user of Developer for System z, it is recommended that you save the related customization as described in “Backing up previously configured files” on page 11.

All references to hlq in this chapter refer to the high level qualifier used during installation of Developer for System z. The installation default is FEK, but this might not apply to your site.

Note: The C/C++ DLL class library CBC.SCLBDLL and the Language Environment (LE) runtime libraries CEE.SCEERUN and CEE.SCEERUN2 must be in LINKLIST.

The LINKLIST requirement can be bypassed by adding a STEPLIB statement to rsed.envvars, see “Customize rsed.envvars, the configuration file for RSE” on page 33.

Set MAXASSIZE in SYS1.PARMLIB(BPXPRMxx)

MAXASSIZE defines the address space (process) region size. Set MAXASSIZE in SYS1.PARMLIB(BPXPRMxx) to 2147483647 or larger.

This value can be checked and set dynamically (until the next IPL) with the following console commands, as described in *MVS System Commands (SA22-7627)*.

1. DISPLAY OMVS,0
2. SETOMVS MAXASSIZE=2147483647

Refer to “Address Space size” on page 79 for more information on other locations where address space sizes can be set or limited.

APF authorize hlq.SFEKLOAD

In order for JES Job Monitor to access JES spool files, module FEJJMON in the hlq.SFEKLOAD load library must be APF authorized.

APF authorizations are defined in SYS1.PARMLIB(PROGxx), if your site followed IBM recommendations. Refer to *MVS Initialization and Tuning Reference (SA22-7592)* for more information on defining APF authorizations.

For testing purposes, APF authorizations can also be given dynamically. These will last until the next IPL of the system. The console command needed will look like this:

```
SETPROG APF,ADD,DSN=hlq.SFEKLOAD,SMS
```

Refer to *MVS System Commands (SA22-7627)* for more information on the SETPROG command.

Customize FEJJCNFG, the JES Job Monitor configuration file

JES Job Monitor is the Developer for System z component that handles all JES connectivity. A configuration file is used to customize certain aspects to meet your site standards.

Note: It is recommended that you copy the sample configuration file to a new data set and customize this copy to avoid overwriting it when applying maintenance.

The sample configuration file is located in:

hlq.SFEKSAMP(FEJJCNFG)

The file consists of a set of environment variable definitions. Comment lines start with the pound sign (#). The sample configuration file is listed in Figure 1.

```
SERV_PORT=6715
CODEPAGE=UTF-8
HOST_CODEPAGE=IBM-1047
TZ=EST5EDT
#LIMIT_VIEW=USERID
LISTEN_QUEUE_LENGTH=5
MAX_DATASETS=32
MAX_THREADS=200
TIMEOUT_INTERVAL=1200
TIMEOUT=3600
AUTHMETHOD=RACF
#_BPXK_SETIBMOPT_TRANSPORT=<tcpip stack name>
```

Figure 1. FEJJCNFG, JES Job Monitor configuration file

The following definitions are required:

SERV_PORT

The port number for JES Job Monitor host server. 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 Developer for System z client users must also change the JES Job Monitor port for this system in Remote Systems View.

Note: Before selecting a port, verify that the port is available on your system with the TSO commands **NETSTAT** and **NETSTAT PORTL**. See “Reserved TCP/IP ports” on page 78 for more information.

Note: When using a version 7.1 or higher client, all communication on this port is confined to your host machine.

CODEPAGE

The workstation 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.

HOST_CODEPAGE

The host codepage. The default is IBM-1047. Change as needed.

TZ Time zone selector. 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 can be found in the *UNIX System Services File System Interface Reference (SA22-7808)*.

LISTEN_QUEUE_LENGTH

The TCP/IP listen queue length. The default is 5. Do not change unless directed to do so by the IBM support center.

MAX_DATASETS

The default is 32. This is the maximum number of spooled output data sets that JES Job Monitor will return to the client (for example, SYSOUT, SYSPRINT, SYS00001, etc.).

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 JES Job Monitor address space.

TIMEOUT_INTERVAL

The default is 1200. Controls how often the server checks for and kills threads which have timed out (see **TIMEOUT**). The value of **TIMEOUT_INTERVAL** specifies the number of seconds between checks.

TIMEOUT

The default is 3600. The length of time, in seconds, before a thread is killed due to lack of interaction with the client. The maximum value is 2147483647.

AUTHMETHOD

The default is RACF, which means that the standard security interface is used. This should not be changed, even if you use a product other than RACF.

The following definitions are optional. If omitted, default values will be used as specified below:

LIMIT_VIEW=USERID

Defines what output the user can view. The default (**LIMIT_VIEW=NOLIMIT**) allows the user to view all JES output. Specifying **USERID** limits the view to output owned by the user.

Note: The only valid settings are **USERID** and **NOLIMIT**.

_BPXK_SETIBMOPT_TRANSPORT=<tcPIP stack name>

Specifies the TCPIP stack to be used in case there are multiple active stacks. The default is TCPIP. <tcPIP stack name> is the requested TCPIP stack name, as defined in the TCPIPJOBNAME statement in the related TCPIP.DATA.

Note: Coding a SYSTCPD DD statement in the JCL does not set the requested stack affinity.

CONCHAR

Specifies 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**.

SUBMITMETHOD=TSO

Submit through TSO. The default (**SUBMITMETHOD=JES**) submits jobs directly into JES. Specifying **SUBMITMETHOD=TSO** causes the job to be submitted via

TSO's **SUBMIT** command. This method allows TSO exits to be invoked; however it has a performance drawback and for that reason it is not recommended.

Note: The only valid settings are TSO and JES.

Note: If SUBMITMETHOD=TSO is specified, then TSO_TEMPLATE must also be defined.

TSO_TEMPLATE=h1q.SFEKSAMP(FEJTSO)

Wrapper JCL for submitting the job via TSO. There is no default value. This statement references the fully qualified member name of the JCL to be used as a wrapper for the TSO submit. See the SUBMITMETHOD statement for more information.

Note: A sample wrapper job is provided in h1q.SFEKSAMP(FEJTSO). Refer to this member for more information on the customization needed.

Note: TSO_TEMPLATE has no effect unless SUBMITMETHOD=TSO is also specified.

Note: The JESNAME definition is no longer required. Since version 7.0, JES Job Monitor automatically detects if your primary JES is JES2 or JES3.

Customize the JES Job Monitor startup JCL

JES Job Monitor is the Developer for System z component that handles all JES connectivity. To do this, a server must be active on the system (either as user job or STC). Follow the JCL customization steps located in h1q.SFEKSAMP(FEJJJCL) to create the JES Job Monitor server according to your site standards.

Note: It is recommended that you copy the sample JCL to a new data set and customize the copy to avoid overwriting it when applying maintenance.

JES Job Monitor tracing

If you need to turn on JES Job Monitor tracing for diagnostic purposes, add "-TV" to the PARM line. Tracing will cause performance degradations and should only be done under the direction of the IBM support center.

```
//JMONITOR EXEC PGM=FEJJMON,TIME=1440,REGION=0M,  
//          PARM=('POSIX(ON),ENVAR("_CEE_ENVFILE=DD:ENVIRON")/ -TV')
```

Tracing can also be controlled by console commands. Assuming that JMON is the job name used, then the first console command listed below will activate tracing and the second one will deactivate it.

1. F JMON,APPL=-TV
2. F JMON,APPL=-TN

Run JES Job Monitor as an STC

JES Job Monitor can run as a user job or started task (STC). The following tasks need to be implemented to define JMON as an STC:

1. The JCL is not required to have a JOB card (preferably not). Most STCs start with a PROC card like the example in Figure 2 on page 19.


```

//JMON      PROC HLQ=FEK,
//          PRM=
//*
//* RD/Z JES JOB MONITOR
//*
//JMONITOR EXEC PGM=FEJJMON,TIME=1440,REGION=0M,
//          PARM=('POSIX(ON),ENVAR("_CEE_ENVFILE=DD:ENVIRON")/&PRM')
//STEPLIB DD DISP=SHR,DSN=&HLQ..SFEEKLOAD
//ENVIRON DD DISP=SHR,DSN=&HLQ..SFEKSAMP(FEJJCNFG)
//SYSPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSIN DD DUMMY
//SYSTCPD DD DISP=SHR,DSN=SYS1.TCPIP.DATA

```

Figure 2. JES Job Monitor JCL

2. The JCL must reside in a system procedure library (SYS1.PROCLIB is the IBM default).

For easy reference it is recommended that the member name matches the procedure name (JMON in the sample above).

3. It is recommended that STCs have a dedicated user ID. For safety reasons, this user ID should be 'protected', by defining the NOPASSWORD keyword (in RACF). This means that RACF will fail any logon attempt that requires a password, but without revoking the user ID.

Use the following sample command to create such a user ID, where userid is the user ID in question, groupid is its default group and uid is the UNIX ID.

```
ADDUSER userid DFLTGRP(groupid) NOPASSWORD OMVS(UID(uid) HOME(/tmp) PROGRAM(/bin/sh))
```

4. STCs must be defined to the security software (e.g. RACF). There are different ways of defining an STC, but using the STARTED class is recommended. To define the STARTED class, your security administrator would issue the following RACF commands;

- a. SETROPTS GENERIC(STARTED)
- b. RDEFINE STARTED ** STDATA(USER(=MEMBER))
- c. SETROPTS CLASSACT(STARTED)
- d. SETROPTS RACLIST(STARTED)

To add JES Job Monitor as an STC, following RACF commands are needed, where jmon is the name of the JCL member and userid is the user ID whose authorities are to be used.

- a. RDEFINE STARTED jmon.* STDATA(USER(userid))
- b. SETROPTS RACLIST(STARTED) REFRESH

Note: By adding the TRUSTED(YES) keyword to the STDATA field [STDATA(USER(userid) TRUSTED(YES)] you can avoid defining the necessary individual permissions for the STC user ID. RACF skips data set security checks for trusted STCs. However, before doing so, ensure that this user ID cannot be abused, by making it protected as described above.

Refer to *Security Server RACF Security Administrator's Guide (SA22-7683)* for more information on started tasks and security.

5. Once previous tasks are completed, JES Job Monitor can be started and stopped as an STC.

The system operator can issue following commands on the console (where JMON is the JES Job Monitor STC name):

- a. Start JMON
- b. stoP JMON
- c. Display A,JMON

If you have the proper authority, you can give these commands from within SDSF, but then the commands must be preceded by a slash ('/'). Refer to *MVS System Commands (SA22-7627)* for more information on console commands.

Server permissions

The user ID assigned to the JES Job Monitor server needs READ access to the Developer for System z load library, hlq.SFEKLOAD.

Verification of JES Job Monitor startup JCL

Start the user job or STC defined in the steps above. If the job ends with return code 66, then hlq.SFEKLOAD is not APF authorized.

JES pool access and security

Conditional pool access

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 the users' access rights are 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 a console defined named JMON (JMON is the only valid name).

For example, your security administrator would issue the following RACF commands:

1. SETROPTS CLASSACT(CONSOLE)
2. RDEFINE CONSOLE JMON UACC(READ)
3. SETROPTS RACLIST(CONSOLE) REFRESH

Use the following RACF commands to permit users to issue JES2 commands only through the JES Job Monitor, where id is the user ID or group ID of Developer for System z users:

1. RDEFINE OPERCMDS JES2.** UACC(NONE)
2. PERMIT JES2.** CLASS(OPERCMDS) ID(id) ACCESS(CONTROL) WHEN(CONSOLE(JMON))
3. SETROPTS RACLIST(OPERCMDS) REFRESH
4. SETROPTS GENERIC(OPERCMDS) REFRESH

CAUTION:

Defining JES commands with universal access NONE in your security software might impact other applications and operations. Test this before activating it on a production system.

Available commands

JES Job Monitor does not provide Developer for System z users full access to the JES spool. Only the commands listed in Table 6 are available. The commands are issued by selecting the appropriate option in the client menu structure (no command prompt). The scope of the commands is further limited by techniques described below.

Table 6. JES Job Monitor console commands

Command	JES2	JES3
Hold	\$Hjobid	*F,J=jobid,H
Release	\$Ajobid	*F,J=jobid,R
Cancel	\$Cjobid	*F,J=jobid,C
Purge	\$Cjobid,P	*F,J=jobid,C

Note: If the user is not the owner of the spool file, only browse is allowed. All actions listed in Table 6 will result in a "Not authorized for job JOBxxx" message.

Without being authorized for these console commands, users will still be able to submit jobs and read job output, if they are given access authorization to spool files.

Limiting access to spool files

To limit users to their own jobs on the JES spool, define the "LIMIT_VIEW=USERID" statement in the JES Job Monitor configuration file (FEJJCNFG). If they need access to a wider range of jobs, but not all, use the standard spool file protection features of your security product, like the JESSP00L class in RACF.

When defining further protection, keep in mind that JES Job Monitor uses **SAPI** (SYSOUT application program interface) to access the spool. This implies that the user needs at least UPDATE access to the spool files, even for browse functionality. This requisite does not apply if you run z/OS v1.7 (z/OS 1.8 for JES3) or higher. Here READ permission is sufficient for browse functionality.

Refer to *Security Server RACF Security Administrator's Guide (SA22-7683)* for more information on JES spool file protection.

Customize ELAXF*, remote build procedures

Developer for System z provides sample JCL procedures that can be used for the JCL generation, remote project builds and remote syntax check features of CICS BMS maps, IMS MFS screens and COBOL, PL/I, Assembler and C/C++ programs.

These procedures allow installations to apply their own standards. This will also ensure that developers use the same procedures with the same compiler options and compiler levels.

SMP/E installs these sample JCL procedures into hlq.SFEKSAMP. If you plan to use these procedures you must:

1. Copy all procedures named ELAXF* into a system procedure library (like SYS1.PROCLIB) that is available to the users.

2. The copied procedures must be customized to reflect naming conventions used on the target system. The required customization is documented within each JCL procedure.

The sample procedures to be copied and customized are listed in Table 7.

Table 7. Sample ELAXF procedures*

Member	Purpose
ELAXFADT	Sample procedure for assembling and debugging High Level assembler programs.
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 preprocess of EXEC SQL statements embedded in COBOL programs.
ELAXFCOT	Sample procedure for doing CICS translation for EXEC CICS statements embedded in COBOL programs.
ELAXFCPC	Sample procedure for doing C compiles.
ELAXFCPP	Sample procedure for doing C++ compiles.
ELAXFGO	Sample procedure for the GO step.
ELAXFLNK	Sample procedure for linking C/C++, COBOL, PLI and High Level Assembler programs.
ELAXFMFS	Sample procedure for creating IMS MFS screens.
ELAXFPLP	Sample procedure for doing DB2 preprocess of EXEC SQL statements embedded in PLI programs.
ELAXFPLT	Sample procedure for doing CICS translation of EXEC CICS statements embedded in PLI programs.
ELAXFPL1	Sample procedure for doing PL/I compiles, integrated CICS translate and integrated DB2 translate.
ELAXFUOP	Sample procedure for generating the UOPT step when building programs that run in CICS or IMS subsystems.

Table 8 holds a checklist for the different product high level qualifiers used in the ELAXF* procedures.

Table 8. ELAXF high level qualifier checklist*

Product	Default HLQ	Value
RD/z	FEK	
CICS	CICSTS31.CICS	
DB2	DSN810	
IMS	IMS	
COBOL	IGY.V3R4M0	
PL/I	IBMZ.V3R6M0	
C/C++	CBC	
LE	CEE	
system LINKLIB	SYS1	

Table 8. ELAXF* high level qualifier checklist (continued)

Product	Default HLQ	Value
system MACLIB	SYS1	

If the ELAXF* procedures cannot be copied into a system procedure library, copy them into a private library and ask the Developer for System z users to add a JCLLIB card (right after the JOB card) to the job properties on the client. Do not customize the sample JCL in the installation data set since maintenance might replace these members and undo your customizations.

```
//MYJOB    JOB <job parameters>
//PROCS JCLLIB ORDER=(hlq.SFEKSAMP)
```

Note: The JCL Generation/Remote project builds and Remote Syntax Check Operations done from a Developer for System z 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 procedure and step names.

Note: The IBM Debug Tool will need to be ordered, installed and configured to support remote debug of assembler, COBOL and PL/I programs. Refer to the *Rational Developer for System z Host Planning Guide (GI11-8296-00)* to know which level of Debug Tool is required for your version of Developer for System z. The installation and customization of this product is not described in this manual.

(Optional) Define an APPC transaction for the TSO Commands service

Defining the APPC transaction has become optional in version 7.1. By default, Developer for System z now uses SCLM Developer Toolkit to provide the TSO Commands service. The configuration of this is done in `rsed.envvars`, which is described in “Customize `rsed.envvars`, the configuration file for RSE” on page 33.

Note: The Transaction Program JCL that is used by APPC to start the TSO Commands service has changed in version 7.1. If you previously defined the TSO Commands service for capturing ISPEXEC output, you must either define a new APPC transaction, or add the NESTMACS keyword to the PARM line, e.g.:

```
// PARM='ISPSTART CMD(%FEKFRSRV TIMEOUT=60) NEWAPPL(ISR) NESTMACS'
```

The TSO Commands service is implemented as an APPC transaction program, FEKFRSRV and must be active for Developer for System z 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.

Note: If you are unfamiliar with APPC, read Appendix F, “Setting up APPC,” on page 113 before continuing with this section.

Preparation

- The following tasks are a prerequisite and must be completed before configuring the TSO Commands Server. The mentioned manuals describe these tasks.
 1. Install, configure, and start VTAM® on your z/OS system. Refer to the *Communications Server bookshelf (F1A1BK61)* for more information.
 2. Install, configure, and start TCP/IP on your z/OS system. Refer to Appendix C, “Setting up TCP/IP,” on page 87 and the *Communications Server bookshelf (F1A1BK61)* for more information.
 3. Configure and start APPC and the APPC transaction scheduler (ASCH) subsystem. Refer to Appendix F, “Setting up APPC,” on page 113 and the *MVS bookshelf (IEA2BK60)* for more information.
- The REXX in Figure 3 can be used to manage APPC through ISPF panels.

```
/* REXX -- APPC administration using ISPF panels */
address ISPEXEC
"LIBDEF ISPMLIB DATASET ID('ICQ.ICQMLIB') STACK"
"LIBDEF ISPPLIB DATASET ID('ICQ.ICQPLIB') STACK"
"LIBDEF ISPSLIB DATASET ID('ICQ.ICQSLIB') STACK"
"LIBDEF ISPTLIB DATASET ID('ICQ.ICQTLIB') STACK"
address TSO "ALTLIB ACT APPLICATION(CLIST)",
            "DSN('ICQ.ICQCCLIB') UNCOND QUIET"
"SELECT CMD(%ICQASRM0) NEWAPPL(ICQ) PASSLIB"
address TSO "ALTLIB DEACT APPLICATION(CLIST) QUIET"
"LIBDEF ISPMLIB"
"LIBDEF ISPPLIB"
"LIBDEF ISPSLIB"
"LIBDEF ISPTLIB"
exit
```

Figure 3. REXX for APPC ISPF panels

Note: Be aware that you can deactivate the APPC transaction with this tool; the transaction is still there but won't accept any connections.

- The definition of the APPC transaction requires skills in various fields of the MVS operating system. Consult with experienced administrators using following checklist before continuing.

Table 9. APPC transaction checklist

Expertise	Required information: <ul style="list-style-type: none"> • Default value • Where to find the answer 	Value
APPC	Data set name of TPDATA <ul style="list-style-type: none"> • Default: SYS1.APPCTP • Value is listed in SYS1.PARMLIB(APPCPMxx) 	
APPC	Transaction name to be used (may not exist) <ul style="list-style-type: none"> • Default: FEKFRSRV • Existing transactions can be queried by selecting "TP Profile Administration" in the APPC ISPF menu 	
APPC	APPC transaction class to be used <ul style="list-style-type: none"> • Default: A • APPC classes are defined in SYS1.PARMLIB(ASCHPMxx) 	
WLM/ SRM	TSO performance group and domain <ul style="list-style-type: none"> • No IBM default (site dependant) 	

Table 9. APPC transaction checklist (continued)

Expertise	Required information:	Value
	<ul style="list-style-type: none"> • Default value • Where to find the answer 	
RACF	Every Developer for System z user has access to an OMVS segment (this is required) <ul style="list-style-type: none"> • No IBM default (site dependant) • TSO RACF command LU userid OMVS will display an existing personal OMVS segment 	
RACF	Every Developer for System z user must have READ access to hlq.SFEKPROC (FEKFRSRV) <ul style="list-style-type: none"> • No IBM default (site dependant) • TSO RACF command LD AUTHUSER DATASET('hlq.SFEKPROC.**') will display users and groups and their access level for the data sets covered by the data set profile 	

Refer to *MVS Planning Workload Management (SA22-7602)* for more information on WLM/SRM management. Refer to *Security Server RACF Security Administrator's Guide (SA22-7683)* for more information on OMVS segments and data set protection profiles.

Note: The APPC transaction class used must have enough APPC initiators to allow one initiator for each user of Developer for System z.

Note: The APPC transaction uses the REXX exec FEKFRSRV, located in hlq.SFEKPROC. Do not change this location if you want possible SMP/E maintenance to be activated automatically.

Implementation

The system programmer or APPC administrator needs to complete the following steps to configure the command facility:

1. Define the scheduling information (class) for the APPC transaction scheduler if you are not using an existing transaction class. Include a definition in SYS1.PARMLIB(ASCHPMxx) for the class to be used by the transaction program FEKFRSRV. This class is used in sample JCL hlq.SFEKSAMP(FEKAPPCC). Therefore the class in FEKAPPCC must match the class defined in SYS1.PARMLIB(ASCHPMxx). For example:

```
CLASSADD
  CLASSNAME(A)
  MAX(20)
  MIN(1)
  MSGLIMIT(200)
```

Note: The TSO Commands service needs the default specifications to be specified in the OPTIONS and TPDEFAULT sections of SYS1.PARMLIB(ASCHPMxx). Refer to Appendix F, "Setting up APPC," on page 113 for more information.

2. Define the APPC transaction that will act as a command server. You can use the sample JCL hlq.SFEKSAMP(FEKAPPCC) to define this transaction. Instructions on how to customize this JCL are located within the JCL. Sample JCL is also provided to display, hlq.SFEKSAMP(FEKAPPCL), or delete, hlq.SFEKSAMP(FEKAPPCX), the transaction.

Note: If you changed the transaction program name (default FEKFRSRV), the new name must be assigned to `_FEKFSCMD_TP_NAME_` in `rsed.envvars`, as described in “Customize `rsed.envvars`, the configuration file for RSE” on page 33.

3. Control the dispatching priority of the `hlq.SFEKPROC(FEKFRSRV)` transaction program by associating FEKFRSRV with a domain and performance group in Workload Manager (WLM). Because FEKFRSRV issues TSO commands, it should be assigned to a TSO performance group.
4. Define a default OMVS segment for the system or an individual one for each user who needs to use remote edit-compile-debug.
5. Give Developer for System z users READ access to `hlq.SFEKPROC(FEKFSERV)`, the TSO Command server.

Note: Setup verification will be done later during the RSE verification. This because RSE implements the TCP/IP connection to the TSO Commands server.

(Optional) Customize ELAXM*, DB2 stored procedure members

The following customization is needed to incorporate the sample DB2 stored procedure (PL/I and COBOL Stored Procedure Builder) into your system. You will need assistance of a WLM administrator and a DB2 administrator to complete these tasks.

After the SMP/E apply, the sample library `hlq.SFEKSAMP` and procedure library `hlq.SFEKPROC` contains the DB2 stored procedure members listed in Table 10.

Table 10. Sample ELAXM DB2 stored procedure members*

Member	Purpose
<code>hlq.SFEKSAMP(ELAXMJCL)</code>	Sample JCL for defining the PL/I and COBOL Stored Procedure Builder to DB2.
<code>hlq.SFEKSAMP(ELAXMSAM)</code>	Sample JCL procedure of the WLM address space for the PL/I and COBOL Stored Procedure Builder.
<code>hlq.SFEKPROC(ELAXMREX)</code>	REXX code for the PL/I and COBOL Stored Procedure Builder.

Note: The DB2 stored procedure uses REXX exec `ELAXMREX`, located in `hlq.SFEKPROC`. Do not change this location if you want possible SMP/E maintenance to be activated automatically.

Note: See Appendix A, “Running multiple instances of Developer for System z,” on page 69 if you want to rename members `ELAXMSAM` or `ELAXMREX`.

1. Copy `ELAXMSAM` to a procedure library (like `SYS1.PROCLIB`) available to the DB2 stored procedure users and customize the JCL as described in its comments. Make sure that the SYSEXEC DD card points to the library where member `ELAXMREX` resides (default `hlq.SFEKPROC`).
2. Use the workload management (WLM) 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 *MVS Planning Workload Management* (SA22-7602) for information on how to do this.

Note: You can create a new application environment in WLM for the PL/I and COBOL Stored Procedure Builder, or you can add the necessary definitions to an existing one.

3. Copy ELAXMJCL to a private JCL library, customize the copy as described in its comments and ask a DB2 administrator to submit the job. Make sure the WLM ENVIRONMENT clause in the CREATE PROCEDURE statement specifies the name of the WLM environment procedure which has been defined for the PL/I and COBOL Stored Procedure Builder (default ELAXMSAM).

(Optional) Customize CICS Bidirectional language support (bidi)

The Developer for System z Enterprise Service Tools (EST) component supports different formats of Arabic and Hebrew interface messages, as well as 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 EST service requestor dynamically specifies bidirectional attributes of interface messages.
- Bidirectional data processing in service 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.
- EST-generated runtime code includes conversion of data between fields in messages that have different bidirectional attributes.

Attention: The load module (both name and coding) has changed compared to previous releases (pre version 7.0). If you have activated a previous bidi release, the old load module(s) must be removed from the CICS RPL concatenation. The default location(s) is/are:

- FEJ.SFEJDLL(FEJBDTRN)
- FEJ.SFEJLMD(FEJBDTRE) (optional, non-PDSE version of FEJBDTRN for Hebrew)

You will need assistance of your CICS administrator to perform following tasks:

1. Place the hlq.SFEKLOAD(FEJBDTRX) program in the CICS RPL concatenation (DD statement DFHRPL). It is recommended that you do this by adding the installation data set to the concatenation so that applied maintenance is automatically available to CICS.

EST bidirectional transformations are performed in the CICS Service Flow Runtime (SFR) environment by the FEJBDTRX module. The FEJBDTRX module is called when bidirectional conversions are needed in EST-generated code (when mapping is generated between fields in messages that have different bidirectional attributes.)

Note: If you do not concatenate the installation directory but copy the FEJBDTRX module into a new/existing data set, keep in mind that this module is a DLL and MUST reside in a PDSE library.

Note: In version 7.1, the hlq.SFEKDLL(FEJBDTRX) program moved to a new load library, hlq.SFEKLOAD(FEJBDTRX).

2. Ask the CICS administrator to set autoinstall to autoactive.

If `autoinstall` is set to `autoInactive`, define the program `FEJBDTRX` to CICS using the `CEDA` command, for example:

```
CEDA DEF PROG(FEJBDTRX) LANG(LE) G(XXX)
```

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

Attention: Bidi code that has been created with previous releases (pre version 7.0) must be `RECOMPILED` in order to use the new `FEJBDTRX` module.

(Optional) Customize Application Deployment Manager (ADM)

The Application Deployment Manager (ADM) provides a common deployment approach and deployment API for all Developer for System z components.

In addition, ADM provides a CICS Resource Definition (CRD) client and (host based) server, which provide the following functions:

1. Allow application developers to define CICS resources in a limited, controlled, and secure fashion.
 - CICS resource definition defaults are supplied by the CICS administrator and stored in the CRD server repository on the host.
 - The Resource Attribute Update Ability is controlled by the CICS administrator (update, protect, hidden).
 - CRD is limited to a small set of resources typically needed by application developers. These include DB2Tran, Doctemplate, File, Mapset, Processtype, Program, TDQ, and Transaction.
 - Authorization to create CICS resource definitions is controlled by RACF or other external security manager.
2. Prevent CICS development access to unauthorized or incorrect VSAM data sets by providing the CICS administrator control over the physical data set name attribute in File definitions. This binding information is stored in the CRD repository on the host.
3. Miscellaneous CRD server development aids.
 - New copy for programs and mapsets
 - List CICS regions
 - DFHRPL list
4. Miscellaneous CRD server Web Service development aids.
 - Perform Pipeline scan to autoinstall `URIMAP` and `WEBSERVICE` definitions
 - Provide Pipeline and `WSBind` pickup directory list
 - Provide `WSDL` file directory list
 - Provide End Point URI list

Developer for System z supplies three transactions that are used by the CRD server when defining and inquiring CICS resources. Sample COBOL source code is provided to allow site specific customizations.

ADMD

For requests that set Web Service and CICS resource defaults. Typically, this is intended for CICS administrators.

ADMI For requests that define, install or uninstall CICS resources.

ADMR

For all other requests that retrieve CICS environmental or resource information.

Refer to *Rational Developer for System z Application Deployment Manager User's Guide* (SC23-7661) for more information on ADM.

The following customization steps are required to activate the CRD server.

Note: You will need the assistance of your CICS administrator to perform some of the following tasks.

Before installing the 7.1.1 version, if you are a previous user of the CRD server, it is recommended that you save the related customization as described in “Backing up previously configured files” on page 11.

Copy the members to be customized from the installation directory to a personal library and customize these copies to avoid overwriting them when applying maintenance:

- hlq.SFEKSAMP(ADNVSAM)
- hlq.SFEKSAMP(ADNPCCSD)

optional (Pipeline Message Handler)

- hlq.SFEKSAMP(ADNCMSGH)
- hlq.SFEKSAMP(ADNSMSGH)

optional (CSD update for non-primary regions)

- hlq.SFEKSAMP(ADNARCSD)

CRD repository

Customize and submit job ADNVSAM to allocate and initialize the CRD server repository VSAM file. Refer to the documentation within ADNVSAM for customization instructions.

It is advised to create a separate repository for each CICS primary connection region. Sharing the repository implies that all related CICS regions will use the same values stored in the repository, and that multiple address spaces will be writing to the VSAM, which must be set up correctly to handle this.

Note: Unless notified otherwise, your current CRD server repository (holding your customized values) can be reused across Developer for System z releases.

CICS primary connection region

The CRD server must be defined to the primary connection region. This is the region that will process Web Service requests from Developer for System z.

- Place the ADM load modules hlq.SFEKLOAD(ADNCRDS) and hlq.SFEKLOAD(ADNCRDR) in the CICS RPL concatenation (DD statement DFHRPL)

of the CICS primary connection region. It is recommended that you do this by adding the installation data set to the concatenation so that applied maintenance is automatically available to CICS

- Customize and submit job ADNPCCSD to update the CICS System Definition (CSD) for the CICS primary connection region. Refer to the documentation within ADNPCCSD for customization instructions
- Use the appropriate CEDA command to install the ADM group for this region, for example:
`CEDA INSTALL GROUP(ADNPCRGP)`

Pipeline Message Handler

The pipeline message handler (ADNSMSGH) is used for security by processing the user ID and password in the SOAP header. ADNSMSGH is referenced by the pipeline configuration file and must therefore be placed into the CICS RPL concatenation. ADNSMSGH is also used to set the transaction ID to ADMD, ADMR, or ADMI depending on the requested operation. You may wish to customize ADNSMSGH to use different transaction ID's.

Using the default:

- Place the hlq.SFEKLOAD(ADNSMSGH) load module in the CICS RPL concatenation (DD statement DFHRPL) of the CICS primary connection region. It is recommended that you do this by adding the installation data set to the concatenation so that applied maintenance is automatically available to CICS.

Customizing ADNSMSGH:

- Customize the sample ADNSMSGH source code (COBOL) for the Pipeline Message Handler.
- Customize and submit job ADNCMSGH to compile the customized ADNSMSGH source. Refer to the documentation within ADNCMSGH for customization instructions.
- Place the resulting ADNSMSGH load module in the CICS RPL concatenation (DD statement DFHRPL) of the CICS primary connection region.

Note: Ensure that the customized ADNSMSGH load module is located before any reference to hlq.SFEKLOAD, otherwise the default one will be used.

(Optional) CICS non-primary connection regions

The CRD server can also be used with one or more non-primary connection regions, which are usually Application Owning Regions (AOR).

Note: It is not necessary to perform these steps if CICSplex® SM is used to manage your CICS environment.

- Place the ADM load module hlq.SFEKLOAD(ADNCRDS) in the CICS RPL concatenation (DD statement DFHRPL) of these non-primary connection regions. It is recommended that you do this by adding the installation data set to the concatenation so that applied maintenance is automatically available to CICS.
- Customize and submit job ADNARCSO to update the CSD for other, non-primary, connection regions. Refer to the documentation within ADNARCSO for customization instructions
- Use the appropriate CEDA command to install the ADM group for these regions, for example:
`CEDA INSTALL GROUP(ADNARRGP)`

Chapter 4. Activating Developer for System z z/OS UNIX components

Before installing the 7.1.1 version, if you are a previous user of Developer for System z, it is recommended that you save the related customization described in “Backing up previously configured files” on page 11.

If you are unfamiliar with z/OS UNIX, it is advised to ask assistance from an experienced z/OS UNIX or other UNIX administrator to perform the tasks listed in this chapter.

The z/OS UNIX commands needed to perform the listed tasks are described briefly for your convenience. Unless noted otherwise, refer to *UNIX System Services Command Reference (SA22-7802)* for more information on these commands.

The tasks described below expect you to be active in z/OS UNIX. This can be done by issuing the TSO command **OMVS**. Use the **exit** command to return to TSO.

MVS provides the possibility to edit z/OS UNIX files using ISPF through the **OEDIT** command. This command can be used both in TSO and OMVS.

Most z/OS UNIX files have the write permission restricted to the owner of the file. This restriction can be bypassed in multiple ways.

- **UID 0**
This is not recommended for “human” user IDs since there are no z/OS UNIX related restrictions.
- **READ** access to the **BPX.SUPERUSER** profile in the **FACILITY** class
Allows the user to become UID 0 through the **su** command. This is the recommended setup.
- **UPDATE** access to the **SUPERUSER.FILESYS** profile in the **UNIXPRIV** class
Allows user to read/write any file, and to read or search any directory. **CONTROL** (or higher) access to this security profile adds writing to any directory to the list of permissions.

Refer to *UNIX System Services Planning (GA22-7800)* to learn more about z/OS UNIX security.

All `/usr/lpp/wd4z/` path statements in this chapter refer to the path used during installation of Developer for System z. The default is `/usr/lpp/wd4z/`, but this might not apply to your site.

Note: Developer for System z is dependant upon TCP/IP having the correct hostname when it is initialized. This implies that the different TCP/IP and Resolver configuration files must be set up correctly. Refer to Appendix C, “Setting up TCP/IP,” on page 87 for more information on the customization necessary.

You can test your TCP/IP configuration with the TSO command **HOMETEST**. See *Communications Server IP System Administrator's Commands (SC31-8781)* for more information on this command.

Note: Developer for System z is dependant upon INETD for setting up the client-host connection. Refer to Appendix D, “Setting up INETD,” on page 93 for more information on INETD.

Note: Remote (host based) actions for z/OS UNIX subprojects require that REXEC or SSH is active on the host.

Note: 31-bit Java is required and all z/OS UNIX users must have EXECUTE and READ access to the Java HFS directories.

Attention: 64-bit Java is NOT supported.

You may find the following publications helpful on understanding z/OS UNIX:

- *ABCs of z/OS System Programming Volume 9 (Redbook SG24-6989)*
- *UNIX System Services Planning (GA22-7800)*
- *UNIX System Services User's Guide (SA22-7801)*

Saving the rsed.envvars configuration file in another directory

It is recommended that you copy the /usr/lpp/wd4z/rse/lib/rsed.envvars file to a new directory (like /etc/wd4z/) and customize the copy to avoid overwriting your customization when applying maintenance. However, when you do this, you must also copy the following files from the installation directory (default /usr/lpp/wd4z/rse/lib/) to the new location:

1. rsed.envvars
2. rsecomm.properties
3. ssl.properties
4. setup.env.zseries
5. server.zseries

Note: Although there is no customization required for the *.zseries files, it is important that you replace previous versions with the current ones. This to keep them in sync with the current rsed.envvars.

The files listed in Table 11 must be copied also if you plan on using the optional features that they are part of:

Table 11. Optional configuration files

file	function
projectcfg.properties	Host based projects See “(Optional) Customize projectcfg.properties, host projects configuration” on page 50
FMIEXT.properties	File Manager integration See “(Optional) Customize FMIEXT.properties, File Manager integration” on page 51
CRASRV.properties	CARMA See Chapter 5, “(Optional) Activating IBM Common Access Repository Manager (CARMA),” on page 53

The following sample commands,

1. `mkdir /etc/wd4z`
2. `cd /usr/lpp/wd4z/rse/lib`
3. `cp rsed.envvars /etc/wd4z`

create a directory named `/etc/wd4z` and copy `rsed.envvars` from the current directory to `/etc/wd4z`. Repeat the copy command for the remaining files.

The result of the copy can be verified with the command `ls /etc/wd4z`, which should give an output similar to this (\$ is the z/OS UNIX prompt):

```
$ ls /etc/wd4z
/etc/wd4z
rsecomm.properties  server.zseries      ssl.properties
rsed.envvars        setup.env.zseries
```

Note: If you want to keep all Developer for System z files in the same (private) HFS, but also want the configuration files placed in `/etc/wd4z`, you can use symbolic links to solve this problem. The following sample commands create a new directory (`/usr/lpp/wd4z/rse/lib/cust`) in the installation HFS and define a symbolic link (`/etc/wd4z`) to it:

1. `mkdir /usr/lpp/wd4z/rse/lib/cust`
2. `ln -s /usr/lpp/wd4z/rse/lib/cust /etc/wd4z`

Customize `rsed.envvars`, the configuration file for RSE

All Developer for System z client connection methods use the variables set in the `rsed.envvars` file, which is located by default in the installation directory, `/usr/lpp/wd4z/rse/lib/`, but could be copied to another directory in the previous step. The sample file provided has the statements listed in Figure 4 on page 34, where comment lines start with a pound sign (#).


```

#####
# (1) required customizations
JAVA_HOME=/usr/lpp/java/J1.4
RSE_HOME=/usr/lpp/wd4z
TZ=EST5EDT
LANG=C
PATH=/bin:/usr/sbin:.
_RSE_CLASS_OPTS=""
_RSE_JAVAOPTS=""
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DTSO_SERVER=APPC"
#####
# (2) required customizations if SCLMDT is used
_CMDSERV_BASE_HOME=/usr/lpp/SCLMDT
_CMDSERV_BASE_LOAD=BWB.SBWBLOAD
_CMDSERV_CONF_HOME=/etc/SCLMDT
_CMDSERV_WORK_HOME=/var/SCLMDT
STEPLIB=NONE
#STEPLIB=$_CMDSERV_BASE_LOAD
_RSE_CMDSERV_OPTS=""
#####
# (3) optional customizations
# RSE_PORTRANGE=8108-8118
# FEKFLOCK_USERID=userid
# FEKFLOCK_JOBNAME=job_name
# FEKFSCMD_TP_NAME=tp_name
# FEKFSCMD_PARTNER_LU=lu_name
#####
# (4) do not change unless directed by IBM support center
RSE_LIB=$RSE_HOME/rse/lib
ICU_LIB=$RSE_HOME/icuc/lib
_CEE_RUNOPTS="ALL31(ON) HEAP(32M,32K,ANYWHERE,KEEP,,) TRAP(ON)"
_CEE_DMPTARG=$HOME/.eclipse/RSE/$RSE_USER_ID
_BPX_SHAREAS=YES
_BPX_SPAWN_SCRIPT=YES
PATH=$JAVA_HOME/bin:$RSE_LIB:$_CMDSERV_BASE_HOME/bin:$PATH
LIBPATH=$JAVA_HOME/bin:$JAVA_HOME/bin/classic:$ICU_LIB:$RSE_LIB:.
CLASSPATH=$RSE_LIB:$RSE_LIB/dstore_core.jar:$RSE_LIB/clientserver.jar
CLASSPATH=$CLASSPATH:$RSE_LIB/dstore_extra_server.jar
CLASSPATH=$CLASSPATH:$RSE_LIB/dstore_miners.jar
CLASSPATH=$CLASSPATH:$RSE_LIB/universalm miners.jar:$RSE_LIB/mvsminers.jar
CLASSPATH=$CLASSPATH:$RSE_LIB/carma.jar:$RSE_LIB/luceneminer.jar
CLASSPATH=$CLASSPATH:$RSE_LIB/mvsluceneminer.jar:$RSE_LIB/cdzminer.jar
CLASSPATH=$CLASSPATH:$RSE_LIB/mvscdzminer.jar:$RSE_LIB/jesminers.jar
CLASSPATH=$CLASSPATH:$RSE_LIB/FAMiner.jar
CLASSPATH=$CLASSPATH:$RSE_LIB/mvsutil.jar:$RSE_LIB/jesutils.jar
CLASSPATH=$CLASSPATH:$RSE_LIB/lucene-1.4.3.jar:$RSE_LIB/cdtparser.jar
CLASSPATH=$CLASSPATH:$RSE_LIB/wdzBidi.jar:$RSE_LIB/fmiExtensions.jar
CLASSPATH=.:$CLASSPATH
_RSE_CMDSERV_OPTS="$&SESSION=SPAWN$_RSE_CMDSERV_OPTS"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DSCLMDT_OPTS='$_RSE_CMDSERV_OPTS'"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DA_PLUGIN_PATH=$RSE_LIB"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Xbootclasspath/p:$RSE_LIB/bidiTools.jar"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Dcom.ibm.cacheLocalHost=true"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -showversion"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Duser.home=$HOME"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Dclient.username=$RSE_USER_ID"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS $_RSE_CLASS_OPTS"
_RSE_SERVER_CLASS=com.ibm.etools.systems.dstore.core.server.Server
_RSE_SERVER_TIMEOUT=120000
#####
# (5) additional environment variables

```

Figure 4. *rse.d.envvars* – RSE configuration file

The following definitions are required:

JAVA_HOME

Java home directory. The default is /usr/lpp/java/J1.4. Change to match your Java installation.

RSE_HOME

RSE home directory. The default is /usr/lpp/wd4z. Change to match your Developer for System z installation.

TZ

Time zone selector. The default is EST5EDT. The default time zone is UTC +5 hours (Eastern Standard Time (EST) Eastern Daylight Savings Time (EDT)). Change to match your time zone. Additional information can be found in the *UNIX System Services File System Interface Reference (SA22-7808)*.

LANG

Specifies the name of the default locale. The default is C. C specifies the POSIX locale and Ja_JP specifies the Japanese locale. Change to match your locale.

PATH

Command path. The default is /bin:/usr/sbin:.. Can be changed if necessary.

_RSE_CLASS_OPTS

Additional Java options for class sharing. The default is "". See "(Optional) Defining extra Java startup parameters with _RSE_*OPTS" on page 38 for more information on this definition.

_RSE_JAVAOPTS

Additional RSE specific Java options. The default is "". See "(Optional) Defining extra Java startup parameters with _RSE_*OPTS" on page 38 for more information on this definition.

Developer for System z uses SCLM Developer Toolkit by default for the TSO Commands service. APPC is used when the following _RSE_JAVAOPTS option is uncommented:

```
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DTSO_SERVER=APPC"
```

Note: Both TSO Commands service methods require more customizations than just the ones in rsed.envvars. The required customizations for the APPC setup are described in "(Optional) Define an APPC transaction for the TSO Commands service" on page 23, those for SCLMDT are described in "Customize ISPF.conf, ISPF configuration file" on page 43.

The following definitions are required if SCLM Developer Toolkit is used, either for the TSO Commands service or when the SCLMDT plug-in is installed in the Developer for System z client.

_CMDSERV_BASE_HOME

SCLM Developer Toolkit home directory. The default is /usr/lpp/SCLMDT. Change to match your SCLM Developer Toolkit installation. This directive is only required when SCLM Developer Toolkit is used (TSO Commands service or client plug-in).

_CMDSERV_BASE_LOAD

SCLM Developer Toolkit load library. The default is BWB.SBWBLOAD. Change to match your SCLM Developer Toolkit installation. This directive is only required when SCLM Developer Toolkit is used (TSO Commands service or client plug-in).

_CMDSERV_CONF_HOME

SCLM Developer Toolkit base configuration directory. The default is /etc/SCLMDT. Change to match your SCLM Developer Toolkit customization. This directive is only required when SCLM Developer Toolkit is used (TSO Commands service or client plug-in).

_CMDSERV_WORK_HOME

SCLM Developer Toolkit base work directory. The default is /var/SCLMDT. Change to match your SCLM Developer Toolkit customization. This directive is only required when SCLM Developer Toolkit is used (TSO Commands service or client plug-in).

STEPLIB

STEPLIB for the RSE server. The default is NONE. Do not change this line, as it acts as a default value.

Developer for System z uses the LINKLIST by default to access the SCLM Developer Toolkit load library. STEPLIB is used when the following STEPLIB directive is uncommented:

STEPLIB=\$_CMDSERV_BASE_LOAD

Note: Using STEPLIB in z/OS UNIX has a negative performance impact, as described in “Avoid use of STEPLIB” on page 63.

_RSE_CMDSERV_OPTS

Additional TSO Commands service specific Java options. The default is "". See “(Optional) Defining extra Java startup parameters with _RSE_*OPTS” on page 38 for more information on this definition. This directive is only required when SCLM Developer Toolkit is used (TSO Commands service or client plug-in).

The following definitions are optional. If omitted, default values will be used.

_RSE_PORTRANGE

Specifies the port range that the RSE server can open for communication with a client. Any port can be used by default. See “(Optional) Defining the PORTRANGE available for RSE” on page 38 for more information on this definition.

_FEKFLOCK_USERID_

User ID to be used by the lock manager. The default is the logon user ID.

_FEKFLOCK_JOBNAME_

Job name to be used by the lock manager. The default is FEKFLK00.

_FEKFSCMD_TP_NAME_

APPC transaction program name. The default value is FEKFRSRV. Uncomment and change this definition if you did not use the default transaction program name when defining the APPC transaction. See “(Optional) Define an APPC transaction for the TSO Commands service” on page 23 for more information on the APPC transaction.

_FEKFSCMD_PARTNER_LU_

Force RSE to use this APPC base LU. See Appendix F, “Setting up APPC,” on page 113 for more information on this definition.

The following definitions are required, and should not be changed unless directed by the IBM support center.

RSE_LIB

RSE library path. The default is \$RSE_HOME/rse/lib. Do not modify.

ICU_LIB

International Components for Unicode (ICU) library path. The default is \$RSE_HOME/icuc/lib. Do not modify.

_CEE_RUNOPTS

Language Environment (LE) runtime options used by the started processes. The default is "ALL31(ON) HEAP(32M,32K,ANYWHERE,KEEP,,) TRAP(ON)". Do not modify.

_CEE_DMPTARG

Language Environment (LE) z/OS UNIX dump location used by the Java virtual Machine (JVM). The default is \$HOME/.eclipse/RSE/\$RSE_USER_ID. Do not modify.

_BPX_SHAREAS

Run foreground processes in the same address space as the shell. The default is YES. Do not modify.

_BPX_SPAWN_SCRIPT

Run shell scripts directly from the spawn() function. The default is YES. Do not modify.

PATH Command path. The default is \$JAVA_HOME/bin:\$RSE_LIB:\$_CMDSERV_BASE_HOME/lib:\$PATH. Do not modify.

LIBPATH

Library path. The default is \$JAVA_HOME/bin:\$JAVA_HOME/bin/classic:\$ICU_LIB:\$RSE_LIB:.. Do not modify.

CLASSPATH

Class path. The default is too long to repeat. Do not modify.

_RSE_CMDSERV_OPTS

Additional TSO Command service specific Java options. The default is "&SESSION=SPAWN\$_RSE_CMDSERV_OPTS". Do not modify.

_RSE_JAVAOPTS

Additional RSE specific Java options. The default is too long to repeat. Do not modify.

_RSE_SERVER_CLASS

Java class for the RSE server. The default is com.ibm.etools.systems.dstore.core.server.Server. Do not modify.

_RSE_SERVER_TIMEOUT

Time out value for the RSE server (waiting on the client) in milliseconds. The default is 120000 (2 minutes). Do not modify.

Note: You can bypass the need of having C/C++ and Language Environment (LE) libraries in LINKLIST by adding the following STEPLIB statement to the END of rsed.envvars (data set names may vary at your site). Be aware however that using STEPLIB in z/OS UNIX has a negative performance impact, as described in "Avoid use of STEPLIB" on page 63.

- If the last STEPLIB directive defined earlier in rsed.envvars equals STEPLIB=NONE
STEPLIB=CEE.SCEERUN:CEE.SCEERUN2:CBC.SCLBDLL
- If the last STEPLIB directive defined earlier in rsed.envvars does not equal STEPLIB=NONE
STEPLIB=\$STEPLIB:CEE.SCEERUN:CEE.SCEERUN2:CBC.SCLBDLL

Note: Symbolic links are allowed when specifying directories in rsed.envvars.

(Optional) Defining the PORTRANGE available for RSE

This is a part of `rsed.envvars` customization that specifies the ports on which the RSE server can communicate with the client. This range of ports has no connection with the RSE daemon or REXEC/SSH ports.

To help understand the port usage, a brief description of RSE's connection process follows:

1. The client connects to host port 4035, INETD RSE daemon service, or host port 512, INETD REXEC service, or host port 22, INETD SSH service.
2. The chosen INETD service creates an RSE process.
3. The RSE process opens a host port for the client to connect. The selection of this port can be configured by the user, either on the client in the subsystem properties tab (this is not recommended) or through the `_RSE_PORTRANGE` definition in `rsed.envvars`.
4. The INETD service returns the port number to the client.
5. The client connects to the host port.

To specify the port range, for the client to communicate with z/OS, uncomment and change the following line in `rsed.envvars`:

```
#_RSE_PORTRANGE=8108-8118
```

Note: Before selecting a port range, verify that the range is available on your system with the **NETSTAT** and **NETSTAT PORTL** commands. See "Reserved TCP/IP ports" on page 78 for more information.

The format of `PORTRANGE` is: `_RSE_PORTRANGE=min-max` (max is non-inclusive; e.g. `_RSE_PORTRANGE=8108-8118` means port numbers from 8108 up to 8117 are usable). The port number used by the RSE server is determined in the following order:

1. If a nonzero port number is specified in the subsystem properties on the client, that the port number is used. If the port is not available connect will fail. This setup is not recommended.
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.

Note: 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 again. This means that the number of ports in the range does not limit the number of users connected concurrently.

(Optional) Defining extra Java startup parameters with `_RSE_*OPTS`

With the different `_RSE_*OPTS` directives, `rsed.envvars` provides the possibility to give extra parameters to Java when it starts the RSE server.

The sample options included in `rsed.envvars` can be activated by uncommenting them.

`_RSE_JAVAOPTS`

`_RSE_JAVAOPTS` defines standard and RSE specific Java options.

_RSE_JAVAOPTS=""

Variable initialization. Do not modify.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -Xquickstart"

Improves startup time by delaying JIT compilation and optimizations. This is at the expense of slightly less efficient compiled executables, which affects long running tasks. Uncomment to activate.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -Xms128m -Xmx128m"

Set initial (Xms) and maximum (Xmx) heap size. System defaults are 1M and 64M respectively. Uncomment and change to enforce the specified heap size values.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -Dfile.encoding=Cp424"

Host codepage selection. Uncomment and change to enforce to use of the specified codepage.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS

-DDENY_PASSWORD_SAVE=true"

Password save option. Uncomment to prevent users from saving their host password on the client. Previously saved passwords will be removed. This option only works with clients version 7.1 and up.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS

-DDSTORE_TRACING_ON=true"

Start dstore tracing. Use only when directed by the IBM support center.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS

-DDSTORE_MEMLOGGING_ON=true"

Start dstore memory tracing. Use only when directed by the IBM support center.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -DTSO_SERVER=APPC"

Use APPC for the TSO Commands service. See "(Optional) Define an APPC transaction for the TSO Commands service" on page 23 for more information.

_RSE_CLASS_OPTS

The **_RSE_CLASS_OPTS** directive defines Java 5.0 (and higher) options needed to share classes between multiple RSE servers. See "Class sharing between JVMs" on page 65 for more information.

_RSE_CLASS_OPTS=""

Variable initialization. Do not modify.

#_RSE_CLASS_OPTS=-Xshareclasses:name=RSE,groupAccess,nonFatal

Java 5.0 and higher only. Enable class sharing. Uncomment to share classes between multiple RSE servers.

#_RSE_CLASS_OPTS="\$_RSE_CLASS_OPTS -Xscmx6m"

Java 5.0 and higher only. Set the size of the shared class cache. System default is 16M.

_RSE_CMDSERV_OPTS

The **_RSE_CMDSERV_OPTS** directives are RSE specific Java options and are only in effect when SCLM Developer Toolkit is used as TSO Commands Server.

```
_RSE_CMDSERV_OPTS=""
```

Variable initialization. Do not modify.

```
#_RSE_CMDSERV_OPTS="$_RSE_CMDSERV_OPTS&ISPPROF=  
&SYSUID..ISPPROF"
```

Use an existing ISPF profile for the TSO Commands Server.

Uncomment and change the data set name to use the specified ISPF profile. &SYSUID. can be used as a substitution for the developer's user ID.

INETD daemon and RSE REXEC/SSH setup

Developer for System z relies on the INETD service to start the Remote Systems Explorer (RSE) Server when a client requests a connection. INETD is a standard z/OS UNIX daemon, who manages other daemons that do the actual work (in this case, starting the RSE server). The configuration of INETD is not a part of Developer for System z customization, but valuable information can be found in Appendix D, "Setting up INETD," on page 93.

Developer for System z supports multiple ways to start the RSE server.

You need to customize at least one way, depending on how your users plan to operate.

- RSE daemon, started by connecting to INETD on port 4035 (default). This is the recommended method because it gives the user more control and security than REXEC.
- REXEC (Remote Execution) Command Server, default port 512, executes a shell script that calls RSE. This method of connecting is usable for proof of concept, but not for long term use. The advantage of using this method is that it requires less installation and configuration work, if REXEC has been setup before (normally done during TCP/IP setup).
- SSH (Secure Shell) Command Server, default port 22, executes a shell script that calls RSE. This method is comparable to REXEC, but SSH uses secure (encrypted) communication with the client.

Note: Remote (host based) actions for z/OS UNIX subprojects require that REXEC or SSH is active on the host.

You can verify that INETD is active with the **ps -e** command (given by an authorized user). The output must contain a reference to INETD, for example (# is the z/OS UNIX prompt):

```
# ps -e  
PID TTY      TIME CMD  
  7 ?          0:00 /usr/sbin/inetd
```

Note: In order for z/OS UNIX servers (like RSE daemon, REXEC and SSH) to support IPv6 connections, tcp6 must be specified for the protocol of the service name in the /etc/inetd.conf file. When tcp6 is defined, IPv4 clients are also supported. /etc/services supports only the tcp keyword, without a number suffix.

INETD RSE daemon set up

1. Modify /etc/services by adding this line:

```
rse      4035/tcp      #Developer for System z RSE
```

rse Service name of the daemon, default is rse (lowercase). The name must match the name that will be used in /etc/inetd.conf

4035/tcp

port and protocol used, the default port is 4035, the protocol must be tcp.

The port used must match the port defined on the client, which is set during the creation of a new z/OS connection.

Note: Before selecting a port, verify that the port is available on your system with the **NETSTAT** and **NETSTAT PORTL** commands. See “Reserved TCP/IP ports” on page 78 for more information.

#Developer for System z RSE

comment, which must start with a pound sign (#)

2. Modify /etc/inetd.conf by adding these two lines. See Appendix D, “Setting up INETD,” on page 93 for continuation rules.

```
rse stream tcp nowait OMVSKERN /usr/lpp/wd4z/rse/lib/fekfrsed
                                rsed -d /usr/lpp/wd4z/rse/lib -t 60
```

rse Service name of the daemon. Default is rse (lowercase). The name must match the name used in /etc/services

stream tcp nowait

INETD specific configuration statements (socket type, protocol, wait flag). Do not modify.

Note: Use the tcp6 keyword instead of tcp to support IPv6 connections.

OMVSKERN

User ID for the RSE daemon process. The default is OMVSKERN. This user ID must be a user ID with a valid OMVS security segment, BPX.DAEMON permission and READ and EXECUTE permission to the Developer for System z installation and configuration directories. Refer to Appendix D, “Setting up INETD,” on page 93 for more details on the requirements for user IDs used for system services.

/usr/lpp/wd4z/rse/lib/fekfrsed

Server program (absolute location of fekrfsed). Default is /usr/lpp/wd4z/rse/lib/fekfrsed

Everything after this INETD argument are server arguments, starting with the server name.

rsed Server name. Do not modify.

-d /usr/lpp/wd4z/rse/lib

Working directory (location of RSE server configuration files). The default is /usr/lpp/wd4z/rse/lib

Note: It is recommended that you copy the customized RSE configuration files to a new directory (like /etc/wd4z/) to avoid overwriting them when applying maintenance. The working directory defined here must reflect this change. For example:

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

The following definitions are optional. If omitted, default values will be used.

-t 60 Timeout option to specify how many seconds the RSE daemon waits

for the RSE server to respond. The default is 60 seconds. The time out for the RSE server waiting on the client is set in `rsed.envvars` and is 2 minutes by default.

3. INETD must be restarted by an authorized user to activate the changes made to the `/etc` files, as described in Appendix D, “Setting up INETD,” on page 93. See the following sample commands (# is the z/OS UNIX prompt):

```
a. # ps -e | grep inetd
50331687 ? 0:00 /usr/sbin/inetd
b. # kill 50331687
c. # _BPX_JOBNAME='INETD' /usr/sbin/inetd
d. # netstat | grep 4035
INETD4 00000B6A 0.0.0.0..4035 0.0.0.0..0 Listen
```

Note: If the BPX.DAEMON profile is defined in the FACILITY class of your security product, and the user (re)starting INETD does not have access to this resource, then the following security warning can be expected for each client connecting to RSE, where IBMUSER is the user ID used to start INETD.

```
ICH408I USER(IBMUSER ) GROUP(SYS1 ) NAME(IBMUSER )
BPX.DAEMON CL(FACILITY)
INSUFFICIENT ACCESS AUTHORITY
ACCESS INTENT(READ ) ACCESS ALLOWED(NONE )
```

INETD REXEC (or SSH) set up

There is no Developer for System z specific setup for using the INETD REXEC (or SSH) Command Server. However, the client needs to know 2 values to start a RSE connection through REXEC/SSH:

- The directory where the `server.zseries` startup script is located.

By default this is the installation directory (`/usr/lpp/wd4z/rse/lib/`). However, `server.zseries` is one of the files that must be copied also if `rsed.envvars` is copied to a different directory, like `/etc/wd4z/`.

- The port that is being used.

A common port used by REXEC is 512. A quick way to check this is the **NETSTAT** command, as shown in the following sample (\$ is the z/OS UNIX prompt):

```
$ netstat | grep 512
INETD4 0000002E 0.0.0.0..512 0.0.0.0..0 Listen
```

To verify this, you can check `/etc/inetd.conf` and `/etc/services` to find the port number used.

1. find the service name (1st word, `exec` in this example) of the `rexecd` server (7th word) in `/etc/inetd.conf`

```
exec stream tcp nowait OMVSKERN /usr/sbin/orexecd rexecd -LV
```

2. Find the port (2nd word, 512 in this example) attached to this service name (1st word) in `/etc/services`

```
exec 512/tcp #REXEC Command Server
```

The same principle applies to SSH. Its common port is 22, and the server name is `sshd`.

Note: Remote (host based) actions for z/OS UNIX subprojects require that REXEC or SSH is active on the host. If REXEC/SSH is not configured to use the default port, the Developer for System z client must define the correct port

for use by z/OS UNIX subprojects. This can be done by selecting the **Window > Preferences... > z/OS Solutions > USS Subprojects > Remote Action Options** preference page.

Customize ISPF.conf, ISPF configuration file

This step is only required when using SCLM Developer Toolkit for the TSO Commands service (this is the default). It is not required when using APPC for the TSO Commands service.

SCLM Developer Toolkit requires the definitions in ISPF.conf to create a valid environment to run ISPF services. The TSO Commands service must be added to this ISPF environment.

ISPF.conf is created during the SCLM Developer Toolkit customization, which is described in *SCLM Developer Toolkit Installation and Customization Guide (SC23-8504)*. The default location is /etc/SCLMDT/CONFIG, but this might not apply to your site.

Add the following lines to ISPF.conf, where hlq equals the high level qualifier used to install Developer for System z (default FEK).

```
*****
* Developer for System z – TSO Commands server
*****
sysexec=hlq.SFEKPROC
```

The result should look like the sample in Figure 5.

```
sysproc=ISP.SISPCLIB
ispmllib=ISP.SISPMENU
isptlib=ISP.SISPTEU
ispplib=ISP.SISPPENU
ispslib=ISP.SISPSLIB
ispllib=BWB.SWBLOAD
*****
* Developer for System z – TSO Commands server
*****
sysexec=FEK.SFEKPROC
```

Figure 5. ISPF.conf - ISPF configuration file

Note: If the sysexec statement is already defined, add the hlq.SFEKPROC data set to the end of it, separating the data set names with a comma (,).

Verify RSE server set up

The Developer for System z installation provides several Installation Verification Programs (IVP) for the RSE server. The IVP scripts are located in the installation directory, default /usr/lpp/wd4z/rse/lib/.

- fekfivpa : “TSO Commands service connection (using APPC)” on page 48
- fekfivpc : “TSO Commands service connection (using SCLMDT)” on page 47
- fekfivpd : “RSE daemon connection” on page 46
- fekfivpj : “JES Job Monitor connection” on page 47
- fekfivpr : “REXEC connection” on page 45
- fekfivps : “REXEC/SSH shell script” on page 46

All sample commands in this section expect that the RSE environment variables are set. This way, the IVP scripts are available through the PATH statement and the location of rsed.envvars is known. Use the **pwd** and **cd** commands to verify and change your current directory to the directory with the customized rsed.envvars. The setup.env.zseries shell script can then be used to set the RSE environment variables, like in the following sample (\$ is the z/OS UNIX prompt):

```
$ pwd
/etc
$ cd /etc/wd4z
$ . ./setup.env.zseries
```

The . ./setup.env.zseries shell script, which resides in the same directory as rsed.envvars, exports the environment variables so that other processes can use them. The first "." (dot) in . ./setup.env.zseries is a z/OS UNIX command to run the shell in the current environment, so that the environment variables set in the shell are effective even after exiting the shell. The second one is referring to the current directory.

Note: If . ./setup.env.zseries is not executed before the fekfivp* scripts, the path to these scripts must be specified when calling them, like in the following sample:

```
/usr/lpp/wd4z/rse/lib/fekfivpr 512 USERID
```

Also, most fekfivp* scripts will ask for the location of the customized rsed.envvars if . ./setup.env.zseries is not executed first.

Note: Some IVP tests use the TCP/IP REXX socket API, which requires that the TCP/IP load library, default TCPIP.SEZALOAD, is in LINKLIST or STEPLIB. The following commands might be necessary to be able to execute these IVP tests (\$ is the z/OS UNIX prompt):

```
$ echo $STEPLIB
none
$ STEPLIB=TCPIP.SEZALOAD

or

$ echo $STEPLIB
SOME.STEPLIB.DATASET
$ STEPLIB=$STEPLIB:TCPIP.SEZALOAD
```

For information on diagnosing RSE connection problems, see Appendix B, "Troubleshooting configuration problems," on page 73 or the technotes on the Developer for System z Support Page <http://www-306.ibm.com/software/awdtools/devzseries/support/>.

Port availability

The JES Job Monitor, REXEC, SSH and RSE daemon port availability can be verified by issuing the **netstat** command. The result should show the ports used by these services, like in the following samples (\$ is the z/OS UNIX prompt):

```
IPv4
$ netstat
MVS TCP/IP NETSTAT CS V1R7      TCP/IP Name: TCPIP      13:57:36
User Id  Conn      Local Socket      Foreign Socket      State
-----  ---
INETD4   00000014  0.0.0.0..22      0.0.0.0..0          Listen
```

```

INETD4 00000030 0.0.0.0..512      0.0.0.0..0      Listen
INETD4 0000004B 0.0.0.0..4035    0.0.0.0..0      Listen
JMON   00000037 0.0.0.0..6715    0.0.0.0..0      Listen

```

IPv6

```

$ netstat
MVS TCP/IP NETSTAT CS V1R7      TCPIP Name: TCPIP      14:03:35
User Id Conn      State
-----
INETD4 00000018 Listen
Local Socket: 0.0.0.0..22
Foreign Socket: 0.0.0.0..0
INETD4 00000046 Listen
Local Socket: 0.0.0.0..512
Foreign Socket: 0.0.0.0..0
INETD4 0000004B Listen
Local Socket: 0.0.0.0..4035
Foreign Socket: 0.0.0.0..0
JMON   00000037 Listen
Local Socket: 0.0.0.0..6715
Foreign Socket: 0.0.0.0..0

```

REXEC connection

Verify the REXEC connection by executing the following command. Replace 512 with the port used by REXEC and USERID by a valid user ID.

```
fekfivpr 512 USERID
```

After prompting you for a password, the command should return the REXEC trace, a timeout warning, the Java version and the RSE server message, like in the following sample (\$ is the z/OS UNIX prompt):

```

$ fekfivpr 512 USERID
Enter password:
$ EZYRC01I Calling function rexec_af with the following:
EZYRC02I Host: CDFMVS08, user USERID, cmd cd /etc/wd4z;export RSE_USER_ID=USERID;./server.zseries -ivp, port 512
EZYRC19I Data socket = 4, Control socket = 6.

```

expect to see time out messages after a successful IVP test

```

java version "1.5.0"
Java(TM) 2 Runtime Environment, Standard Edition (build pmz31dev-20070201 (SR4))
IBM J9 VM (build 2.3, J2RE 1.5.0 IBM J9 2.3 z/OS s390-31 j9vmmz3123-20070201 (JIT enabled)
J9VM - 20070131_11312_bHdSMr
JIT - 20070109_1805ifx1_r8
GC - 200701_09)
JCL - 20070126

```

```

Server Started Successfully
1272
Server running on: CDFMVS08

```

Note: If you don't get any Java and RSE server output, the INETD region size is probably too small (must be 2096128 or larger if started from a TSO/OMVS shell session, or region size 0 for BPXBATCH).

Note: You can test the shell script used by REXEC separately, as described in the next IVP test, "REXEC/SSH shell script" on page 46.

Note: The server is started without a client trying to connect, so it will time out (after 5 seconds). This results in a Java stack trace (25+ lines) that looks like the following sample:

```
$ java.net.SocketTimeoutException: Accept timed out
    at java.net.PlainSocketImpl.socketAccept(Native Method)
    at java.net.PlainSocketImpl.accept(PlainSocketImpl.java:384)
    at java.net.ServerSocket.implAccept(ServerSocket.java:471)
    at java.net.ServerSocket.accept(ServerSocket.java:442)
    at com.ibm.etools.systems.dstore.core.server.ConnectionEstablisher.
...

```

REXEC/SSH shell script

This IVP test can be skipped if the previous test outlined in, “REXEC connection” on page 45, completed successfully.

Verify the shell script used by the REXEC and SSH connection by executing the following command:

```
fekfivps
```

The command should return a timeout warning, the Java version and the RSE server message, like in the following sample (\$ is the z/OS UNIX prompt):

```
$ fekfivps
$ java version "1.5.0"
```

expect to see time out messages after a successful IVP test

```
Java(TM) 2 Runtime Environment, Standard Edition (build pmz31dev-20070201 (SR4))
IBM J9 VM (build 2.3, J2RE 1.5.0 IBM J9 2.3 z/OS s390-31 j9vmz3123-20070201 (JIT enabled))
J9VM - 20070131_11312_bHdSMr
JIT - 20070109_1805ifx1_r8
GC - 200701_09)
JCL - 20070126
```

```
Server Started Successfully
1751
Server running on: CDFMVS08$
```

Note: If you don’t get any output, your (TSO) region size is probably too small (must be 2096128 or larger).

Note: The server is started without a client trying to connect, so it will time out (after 5 seconds). This results in a Java stack trace (25+ lines) that looks like the following sample:

```
$ java.net.SocketTimeoutException: Accept timed out
    at java.net.PlainSocketImpl.socketAccept(Native Method)
    at java.net.PlainSocketImpl.accept(PlainSocketImpl.java:384)
    at java.net.ServerSocket.implAccept(ServerSocket.java:471)
    at java.net.ServerSocket.accept(ServerSocket.java:442)
    at com.ibm.etools.systems.dstore.core.server.ConnectionEstablisher.
...

```

RSE daemon connection

Verify the RSE daemon connection by executing the following command. Replace 4035 with the port used by the RSE daemon and USERID by a valid user ID.

```
fekfivpd 4035 USERID
```

After prompting you for a password the command should return an output like in this sample (\$ is the z/OS UNIX prompt):

```
$ fekfivpd 4035 USERID
Password:
SSL is disabled
connected
8108
570655399
Success
```

Note: When testing an SSL enabled connection, verify that you specified the correct port if you get this error message: gsk_secure_socket_init() failed: Socket closed by remote partner

JES Job Monitor connection

Verify the JES Job Monitor connection by executing the following command. Replace 6715 with the JES Job Monitor port number.

```
fekfivpj 6715
```

The command should return the JES Job Monitor acknowledge message, like in the following sample (\$ is the z/OS UNIX prompt):

```
$ fekfivpj 6715
Waiting for JES Job Monitor response...
ACKNOWLEDGE01v03
```

Success

TSO Commands service connection (using SCLMDT)

This IVP test is only needed if you use SCLM Developer Toolkit, either for the TSO Commands service or client plug-in.

Verify the connection to the TSO Commands server using SCLM Developer by executing the following command.

```
fekfivpc
```

The command should return the result of SCLM Developer Toolkit related checks (variables, HFS modules, REXX runtime, starting and stopping TSO/ISPF session), like in the following sample (\$ is the z/OS UNIX prompt):

```
$ fekfivpc
-----
Host install verification for RSE
Review IVP log messages from HOST below :
-----

RSE connection and base TSO/ISPF session initialization check only

*** CHECK : ENVIRONMENT VARIABLES - key variables displayed below :

Server PATH          = /usr/lpp/java/J5.0/bin:/usr/lpp/wd4z/rse/lib:/usr/lpp/SCLM
DT/bin:/bin:/usr/sbin.

STEPLIB              = BWB.SWBLOAD

_CMDSERV_BASE_HOME   = /usr/lpp/SCLMDT
_CMDSERV_CONF_HOME   = /etc/SCLMDT
_CMDSERV_WORK_HOME   = /var/SCLMDT
-----

*** CHECK : HFS MODULES
Checking install Directory : /usr/lpp/SCLMDT
Checking for BWB modules in /bin directory
RC=0
MSG: SUCCESSFUL
```

```
-----
*** CHECK : REXX RUNTIME ENVIRONMENT
RC=0
MSG: SUCCESSFUL
```

```
-----
*** CHECK : TSO/ISPF INITIALIZATION
( TSO/ISPF session will be initialized )
RC=0
MSG: SUCCESSFUL
```

```
-----
*** CHECK: Shutting down TSO/ISPF IVP session
RC=0
MSG: SUCCESSFUL
```

```
-----
Host installation verification completed successfully
-----
```

Note: If any of the SCLMDT checks fail, more detailed information will be shown.

fekfivpc has several optional, non-positional, parameters:

-file fekfivpc can produce large amounts of output (hundreds of lines). The -file parameter sends this output to a file, home/.eclipse/RSE/USERID/fekfivpc.log, where home is the home path defined in your OMVS segment (or the default OMVS segment if you do not have and OMVS segment) and USERID is your user ID (uppercase).

-plugin
By default, fekfivpc only checks the functions needed for the TSO Commands service. The -plugin parameter adds extra tests for the SCLMDT client plug-in.

-debug
The -debug parameter creates detailed test output. Do not use this option unless directed by the IBM support center.

TSO Commands service connection (using APPC)

Do not execute this procedure if you have not set up APPC for the TSO Commands service.

Verify the connection to the TSO Command server (using APPC) by executing the following command. Replace USERID with a valid user ID.

```
fekfivpa USERID
```

After prompting you for a password, the command should return the APPC conversation, like in the following sample (\$ is the z/OS UNIX prompt):

```
$ fekfivpa USERID
Enter password:
20070607 13:57:18.584060 /usr/lpp/wd4z/rse/lib/fekfscmd: version=Oct 2003
20070607 13:57:18.584326 Input parms: 1.2.3.4 * NOTRACE USERID *****
20070607 13:57:18.585132 TP_name set via envvar: FEKFRSRV
20070607 13:57:18.586800 APPC: Allocate succeeded
20070607 13:57:18.587022 Conversation id is 0DDBD3F800000000
20070607 13:57:18.587380 APPC: Set Send Type succeeded
20070607 13:57:26.736674 APPC: Confirm succeeded
20070607 13:57:26.737027 Req to send recd value is 0
20070607 13:57:26.737546 APPC: SEND_DATA return_code = 0
20070607 13:57:26.737726 request_to_send_received = 0
```

```

20070607 13:57:26.737893 Send Data succeeded
20070607 13:57:26.738169 APPC: Set Prepare to Receive type succeeded
20070607 13:57:26.738580 APPC: Prepare to Receive succeeded
20070607 13:57:26.808899 APPC: Receive data
20070607 13:57:26.809122 RCV return_code = 0
20070607 13:57:26.809270 RCV data_received= 2
20070607 13:57:26.809415 RCV received_length= 29
20070607 13:57:26.809556 RCV status_received= 4
20070607 13:57:26.809712 RCV req_to_send= 0
20070607 13:57:26.809868 Receive succeeded
:IP: 0 9.42.112.75 1674 50246
20070607 13:57:26.810533 APPC: CONFIRMED succeeded

```

For information on diagnosing RSE connection problems, see Appendix B, “Troubleshooting configuration problems,” on page 73 or the technotes on the Developer for System z Support Page <http://www-306.ibm.com/software/awdtools/devzseries/support/>.

(Optional) Customize `ssl.properties`, RSE SSL configuration

All Developer for System z client connection methods use the Secure Socket Layer (SSL) variables set in the `ssl.properties` file, which is located by default in the installation directory, `/usr/lpp/wd4z/rse/lib/`. However, `ssl.properties` is one of the files that must be copied also if `rsd.envvars` is copied to a different directory, like `/etc/wd4z/`. The sample file provided has the statements listed in Figure 6, where comment lines start with a pound sign (#).

```

# Specify this property as true to enable SSL
enable_ssl=false

#####
# Daemon Properties
# The key database file and password need to be specified for
# daemon to use.
# The key label need to be specified if not default key.
#daemon_keydb_file=
#daemon_keydb_password=
#daemon_key_label=

#####
# Server Properties
# The keystore file and password need to be specified for the
# server to use.
#server_keystore_file=
#server_keystore_password=

```

Figure 6. `ssl.properties` – SSL configuration file

The daemon and server properties only need to be set if you enable SSL. Refer to Appendix E, “Setting up SSL,” on page 101 for more information on SSL setup.

(Optional) Customize `rsecomm.properties`, RSE trace configuration

All Developer for System z client connection methods use the variables set in the `rsecomm.properties` file, which is located by default in the installation directory, `/usr/lpp/wd4z/rse/lib/`. However, `rsecomm.properties` is one of the files that must be copied also if `rsd.envvars` is copied to a different directory, like `/etc/wd4z/`. The sample file provided has the statements, listed in Figure 7 on page 50, where comment lines start with a pound sign (#).

```
# server.version - DO NOT MODIFY!
server.version=5.0.0

# Logging level
# 0 - Log error messages
# 1 - Log error and warning messages
# 2 - Log error, warning and info messages
# 3 - Log error, warning, info and debug messages
debug_level=1

# Log location
# Log_To_StdOut
# Log_To_File
log_location=Log_To_File
```

Figure 7. rsecomm.properties – Logging configuration file

When selecting `log_location=Log_To_File` (the default), the logging is written to `home/.eclipse/RSE/USERID/rsecomm.log`, where `home` is the home path defined in the user's OMVS segment (or the default OMVS segment if the user does not have and OMVS segment) and `USERID` is the logon user ID (uppercase).

Note: The `debug_level` definition also controls the logging level of the other log files that can be found in this directory.

Attention: Changing these settings can cause performance degradations and should only be done under the direction of the IBM support center.

(Optional) Customize projectcfg.properties, host projects configuration

z/OS Projects can be defined individually through the z/OS Projects perspective on the client or can be defined centrally on the host and propagated to the client on a per user basis. These "host based projects" look and function exactly like projects defined on the client except that their structure, members, and properties cannot be modified by the client and they are only accessible when connected to the host.

The location of the project definitions is defined in `projectcfg.properties`, which is located by default in the installation directory, `/usr/lpp/wd4z/rse/lib/`. However, `projectcfg.properties` is one of the files that must be copied also if `rsd.envvars` is copied to a different directory, like `/etc/wd4z/`.

The sample file provided has the statements listed in Figure 8, where comment lines start with a pound sign (#).

```
#
# host based projects – root configuration file
#
# WSED-VERSION – do not modify !
WSED-VERSION=7.0.0.0
# specify the location of the host based project definition files
PROJECT-HOME=/var/wd4z/projects
```

Figure 8. projectcfg.properties – Host based projects configuration file

The only variable to be changed is `PROJECT-HOME`. Its value, default `/var/wd4z/projects`, is the base directory for the project definitions.

Note: In order to activate host based projects, a `project.instance` file must exist in `/var/wd4z/projects/USERID`, where `/var/wd4z/projects` is the location of the project definition files and `USERID` is the user ID with which the developer logs on.

For more information on host based projects, see the white paper *Host Based Projects in WebSphere Developer for System z version 7.0* in the Developer for System z internet library, <http://www-306.ibm.com/software/awdtools/devzseries/library/>

(Optional) Customize FMEXT.properties, File Manager integration

Developer for System z supports direct access from the client to a limited set of IBM File Manager for z/OS functions. IBM File Manager for z/OS provides comprehensive tools for working with MVS data sets, z/OS UNIX files, DB2, IMS and CICS data. These tools include the familiar browse, edit, copy and print utilities found in ISPF, enhanced to meet the needs of application developers. In the current version of Developer for System z, only browse/edit of MVS data sets (including VSAM KSDS and ESDS) is supported.

Note that the IBM File Manager for z/OS product must be ordered, installed and configured separately. Refer to *Rational Developer for System z Host Planning Guide (GI11-8296-00)* to know which level of File Manager is required for your version of Developer for System z. The installation and customization of this product is not described in this manual.

The File Manager definitions needed by Developer for System z are stored in `FMEXT.properties`, which is located by default in the installation directory, `/usr/lpp/wd4z/rse/lib/`. However, `FMEXT.properties` is one of the files that must be copied also if `rse.envvars` is copied to a different directory, like `/etc/wd4z/`.

The sample file provided has the statements listed in Figure 9, where comment lines start with a pound sign (#).

```
# File Manager Integration (FMI) Extension properties
#
startup.script=/usr/lpp/wd4z/rse/lib/fmiSub
startup.port=1957
startup.range=100
startup.fmload=FMN.SFMNMOD1
startup.jobcard1=//JOB CARD JOB <job parameters>
startup.jobcard2=//*
startup.jobcard3=//*
startup.sysout=*
```

Figure 9. `FMEXT.properties` – File Manager configuration file

startup.script

Absolute location of `fmiSub`, the FMI server startup script. The default value is `/usr/lpp/wd4z/rse/lib/fmiSub`.

startup.port

First port used for communication between the FMI server and the RSE server, which relays the information to the client. The default port is 1957. Communication on this port is confined to your host machine.

Note: Before selecting a port, verify that the port is available on your system with the `NETSTAT` and `NETSTAT PORTL` commands. See “Reserved TCP/IP ports” on page 78 for more information.

startup.range

Range of ports, starting at `startup.port`, which will be used for FMI server communication. The default is 100. For example, when using the defaults, port 1957 until 2056 (inclusive) can be used by the FMI server.

startup.fmload

Absolute location of the File Manager load library. The default value is `FMN.SFMNMOD1`. Do not use quotes (') to make the data set name absolute, a prefix is not added.

Note: File Manager has multiple load libraries. The one that must be referenced in this configuration file is `SFMNMOD1`.

startup.jobcard1**startup.jobcard2****startup.jobcard3**

Jobcard information for the FMI server. The default values are `//JOB CARD JOB <job parameters>, /*` and `/*`. The job name will be replaced with `FEK<port>` to ensure uniqueness.

startup.sysout

Sysout class for the FMI server. The default value is `*`.

Chapter 5. (Optional) Activating IBM Common Access Repository Manager (CARMA)

Common Access Repository Manager (CARMA) (FMID: HCMA710) is a productivity aid for developers who are creating APIs for Software Configuration Managers (SCM). In turn, these API's can be used by applications (for example, Developer for System z) to access the SCMs.

Before installing the 7.1.1 version, if you are a previous user of CARMA, it is recommended that you save the related customization as described in "Backing up previously configured files" on page 11.

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

1. Configure the CARMA server on your z/OS host (requires actions in MVS and z/OS UNIX)
2. (Optional) Configure the sample RAMs
3. (Optional) Restrict access to the initialization files and VSAM clusters. Under most circumstances, only System Administrators and CARMA RAM developers will need to write to these files, while other users will only require read access.

Note: Repository Access Managers (RAMs) are user-written API's to interface with z/OS Software Configuration Managers (SCMs).

Refer to Table 2 on page 1 for a list of manuals that provide more information on CARMA and how to use it.

The user can control the amount of trace info CARMA generates by setting Trace Level in the properties tab of the CARMA connection on the client. The choices for Trace Level are:

- Disable Logging
- Error Logging
- Warning Logging
- Informational Logging
- Debug Logging

The default value is

Error Logging

Refer to "Location of log files" on page 73 for more information on log file locations.

Customizing the CARMA MVS components

All references to hlq in this section refer to the high level qualifier used during installation of CARMA. The installation default is CRA, but this might not apply to your site.

Note: In version 7.1, new messages have been added to the CARMA message VSAM, CRAMSG. It is advised to update your previous message VSAM. Also, there was a name change for the sample skeleton RAM in version 7.1, which

results in a change to the CARMA configuration VSAM, CRADEF. Updating this VSAM is only necessary if you plan on using the skeleton RAM.

Follow these steps to configure your MVS host:

1. Copy the members to be customized from the installation directory to a personal library and customize these copies to avoid overwriting them when applying maintenance.
 - hlq.SCRACLST(CRASUBMT)
 - hlq.SCRASAMP(CRA\$VDEF)
 - hlq.SCRASAMP(CRA\$VMSG)
 - hlq.SCRASAMP(CRA\$VSTR)
2. Customize the hlq.SCRACLST(CRASUBMT) CLIST. Refer to the documentation within CRASUBMT for customization instructions. The CRASUBMT CLIST submits a CARMA server.

Note: You can optionally change CARMA's timeout value by modifying the PROC 1 PORT TIMEOUT(420) line in hlq.SCRACLST(CRASUBMT) CLIST. The timeout value is the number of seconds CARMA will wait for the next command from the client. Setting a value of 0 results in the default timeout value, currently 420 seconds (7 minutes).

3. Customize and submit the hlq.SCRASAMP(CRA\$VDEF) JCL. Refer to the documentation within CRA\$VDEF for customization instructions.

Note: You can use the CRA\$VDEF JCL to update the CRADEF VSAM cluster (CARMA configuration) 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 *Rational Developer for System z Common Access Repository Manager Developer's Guide (SC23-7660)* for more information on defining this sequential data set.

4. Customize and submit the hlq.SCRASAMP(CRA\$VMSG) JCL. Refer to the documentation within CRA\$VMSG for customization instructions. CRA\$VMSG creates and primes the CARMA message VSAM, CRAMSG.
5. Customize and submit the hlq.SCRASAMP(CRA\$VSTR) JCL. Refer to the documentation within CRA\$VSTR for customization instructions.

Note: You can use the CRA\$VSTR JCL to update the CRASTRS VSAM cluster (CARMA custom information) 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 *Rational Developer for System z Common Access Repository Manager Developer's Guide (SC23-7660)* for more information on defining this sequential data set.

Customizing the CARMA z/OS UNIX components

If you are unfamiliar with z/OS UNIX, it is advised to ask assistance from an experienced z/OS UNIX or other UNIX administrator to perform the tasks listed in this section.

The z/OS UNIX commands needed to perform the listed tasks are described briefly for your convenience. Unless noted otherwise, refer to *UNIX System Services Command Reference (SA22-7802)* for more information on these commands.

- The tasks described below expect you to be active in z/OS UNIX. This can be done by issuing the TSO command **OMVS**. Use the **exit** command to return to TSO.
- MVS provides the possibility to edit z/OS UNIX files using ISPF through the **OEDIT** command. This command can be used both in TSO and OMVS.

All `/usr/lpp/wd4z/` path statements in this section refer to the path used during installation of Developer for System z, not CARMA. The default is `/usr/lpp/wd4z/`, but this might not apply to your site.

Follow these steps to configure the z/OS UNIX CARMA components, which are installed during the IBM Rational Developer for System z installation (FMID: HHOP710):

1. The `CRASRV.properties` configuration file must reside in the same directory as the customized `rsed.envvars` file. Both files reside by default in the install directory (default path `/usr/lpp/wd4z/rse/lib/`). But as described in “Saving the `rsed.envvars` configuration file in another directory” on page 32, it is advised to copy them to another directory to avoid overwriting them when applying maintenance. In the samples used in this book, this directory is `/etc/wd4z/`.

```
cp /usr/lpp/wd4z/rse/lib/CRASRV.properties /etc/wd4z
```

2. The sample configuration file `CRASRV.properties`, consists of a set of environment variable definitions. The sample configuration file must be changed to match your site standards and contains the statements listed in Figure 10, where comment lines start with a pound sign (#).

```
# CARMA configuration option
#
port.start=5227
port.range=100
startup.script.name=/usr/lpp/wd4z/rse/lib/rexxsub
clist.dsname='hlq.SCRACLST(CRASUBMT)'
```

Figure 10. *CRASRV.properties* – CARMA configuration file

port.start

First port used for communication between CARMA MVS and z/OS UNIX components. The default port is 5227. Communication on this port is confined to your host machine.

Note: Before selecting a port, verify that the port is available on your system with the **NETSTAT** and **NETSTAT PORTL** commands. See “Reserved TCP/IP ports” on page 78 for more information.

port.range

Range of ports, starting at `port.start`, which will be used for CARMA Server communication. The default is 100. For example, when using the defaults, port 5227 until 5326 (inclusive) can be used by CARMA.

startup.script.name

Defines the absolute path of the REXX submit script `rexxsub`. The default is `/usr/lpp/wd4z/rse/lib/rexxsub`. This REXX exec will trigger the execution of the `CRASUBMT CLIST` in MVS.

clist.dsname

Defines the location of the `CRASUBMT CLIST`, using MVS referencing conventions. With apostrophes (') it is an absolute location, without the

user's prefix precedes the data set name provided. The default is 'hlq.SCRACLST(CRASUBMT)'. The CARMA SMP/E installation which creates CRASUBMT, uses CRA as default value for hlq. This CLIST will start a CARMA server when opening a connection.

Note: In Developer for System z version 7.0, the CLIST data set and member name has moved from rexsub (variable DSNAME) into CRASRV.properties, eliminating the need to customize rexsub. Leave rexsub in the install directory if you want possible SMP/E maintenance to be activated automatically.

(Optional) Activating the sample Repository Access Managers (RAMs)

Repository Access Managers (RAMs) are user-written API's to interface with z/OS Software Configuration Managers (SCMs). Follow the instructions in the sections below for the sample RAMs you want to activate.

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.

Note: All references to hlq in this section refer to the high level qualifier used during installation of CARMA. The installation default is CRA, but this might not apply to your site.

Refer to the *Rational Developer for System z Common Access Repository Manager Developer's Guide (SC23-7660)* for more information on the sample RAMs and sample source code provided.

Activating the SCLM RAM

1. Copy the members to be customized from the installation directory to a personal library and customize these copies to avoid overwriting them when applying maintenance.
 - hlq.SCRASAMP(CRA#VSLM)
 - hlq.SCRASAMP(CRA#ASLM)
 - hlq.SCRACLST(CRASUBMT)
2. Customize and submit the hlq.SCRASAMP(CRA#VSLM) JCL. Refer to the documentation within CRA#VSLM for customization instructions. CRA#VSLM creates and primes the SCLM RAM message VSAM.
3. Uncomment the CRARAM2 DD statement in CRASUBMT and provide the data set name of the SCLM RAM message VSAM. Note that CRASUBMT has been customized earlier in "Customizing the CARMA MVS components" on page 53.
4. Customize the hlq.SCRASAMP(CRA#ASLM) JCL. Refer to the documentation within CRA#ASLM for customization instructions. CRA#ASLM allocates data sets needed by SCLM RAM clients.

Note: Each user must submit CRA#ASLM once prior to using CARMA with the SCLM RAM. Failing to do so will result in an allocation error.

Activating the PDS RAM

1. Copy the members to be customized from the installation directory to a personal library and customize these copies to avoid overwriting them when applying maintenance.
 - hlq.SCRASAMP(CRA#VPDS)
 - hlq.SCRACLST(CRASUBMT)
2. Customize and submit the hlq.SCRASAMP(CRA#VPDS) JCL. Refer to the documentation within CRA#VPDS for customization instructions. CRA#VPDS creates and primes the PDS RAM message VSAM.
3. 3. Uncomment the CRARAM1 DD statement in CRASUBMT and provide the data set name of the PDS RAM message VSAM. Note that CRASUBMT has been customized earlier in “Customizing the CARMA MVS components” on page 53.

Activating the skeleton RAM

1. Copy the members to be customized from the installation directory to a personal library and customize these copies to avoid overwriting them when applying maintenance.
 - hlq.SCRASAMP(CRA#CRAM)
 - hlq.SCRACLST(CRASUBMT)
2. Customize and submit the hlq.SCRASAMP(CRA#CRAM) JCL. Refer to the documentation within CRA#CRAM for customization instructions. CRA#CRAM compiles the skeleton RAM.
3. Add the load library holding the compiled skeleton RAM module, CRARAMSA, to the STEPLIB DD of CRASUBMT. Note that CRASUBMT has been customized earlier in “Customizing the CARMA MVS components” on page 53.

Chapter 6. (Optional) Activating IBM Software Configuration and Library Manager (SCLM) Developer Toolkit

IBM Software Configuration and Library Manager (SCLM) Developer Toolkit (FMID: HSD3310) provides the tools needed to extend the capabilities of SCLM to the client. SCLM itself is a host based source code manager that is shipped as part of ISPF.

The SCLM Developer Toolkit, which is shipped together with Developer for System z, is an Eclipse based plug-in that interfaces to SCLM and provides for access to all SCLM processes for legacy code development as well as support for full Java and J2EE development on the workstation with synchronization to SCLM on the mainframe including building, assembling and deployment of the J2EE code from the mainframe.

Refer to Table 2 on page 1 for a list of manuals that provide more information on SCLM Developers Toolkit and how to install, customize and use it.

Chapter 7. Developer for System z client considerations

Users of the Developer for System z client must know the result of certain host customizations, like TCP/IP port numbers, for the client to work properly. Use the checklist in Table 12 to gather the information needed.

Table 12. Developer for System z client checklist

Customization	Value
JES Job Monitor server port number (default 6715): See SERV_PORT in “Customize FEJJCNFG, the JES Job Monitor configuration file” on page 16.	
Location of the ELAXF* procedures if they are not in a system procedure library: See note on JCLLIB in “Customize ELAXF*, remote build procedures” on page 21.	
Procedure and/or step names of the ELAXF* procedures if they were changed: See note on changing them in “Customize ELAXF*, remote build procedures” on page 21.	
DB2 stored procedure name (default ELAXMSAM): See information on DB2 stored procedures in Appendix A, “Running multiple instances of Developer for System z,” on page 69.	
Location of the DB2 stored procedure if it is not in a system procedure library: See “(Optional) Customize ELAXM*, DB2 stored procedure members” on page 26.	
Use DAEMON, REXEC or SSH connection method for RSE: See “INETD daemon and RSE REXEC/SSH setup” on page 40.	
RSE daemon TCP/IP port number (default 4035): See “INETD RSE daemon set up” on page 40.	
Path to the server.zseries shell script for REXEC/SSH (default /usr/lpp/wd4z/rse/lib, advised /etc/wd4z): See “INETD REXEC (or SSH) set up” on page 42.	
REXEC or SSH port number (default 512 or 22, respectively): See “INETD REXEC (or SSH) set up” on page 42. Note: Remote (host based) actions for z/OS UNIX subprojects require that REXEC or SSH is active on the host.	
Location of the CRA#ASLM JCL for CARMA SCLM RAM data set allocations: See note on CRA#ASLM in “Activating the SCLM RAM” on page 56.	

Chapter 8. Performance considerations

z/OS is a highly customizable operating system, and (sometimes small) system changes can have a huge impact on the overall performance. This chapter highlights some of the changes that can be made to improve the performance of Developer for System z.

Refer to the *MVS Initialization and Tuning Guide (SA22-7591)* and *UNIX System Services Planning (GA22-7800)* for more information on system tuning.

Use zFS file systems

zFS (zSeries File System) and HFS (Hierarchical File System) are both UNIX file systems that can be used in a z/OS UNIX environment. However, zFS provides the following features and benefits:

- Performance gains in many customer environments when accessing files approaching 8K in size that are frequently accessed and updated. The access performance of smaller files is equivalent to that of HFS.
- Read-only cloning of a file system in the same data set. The cloned file system can be made available to users to provide a read-only point-in-time copy of a file system. This is an optional feature that is available only in a non-sysplex environment.
- zFS is the strategic z/OS UNIX file system. The HFS functionality has been stabilized, and enhancements to the file system will be for zFS only.

Refer to *UNIX System Services Planning (GA22-7800)* to learn more about zFS.

Avoid use of STEPLIB

Each z/OS UNIX process that has a STEPLIB that is propagated from parent to child or across an exec will consume about 200 bytes of ECSA (Extended Common Storage Area). If no STEPLIB environment variable is defined, or when it is defined as STEPLIB=CURRENT, z/OS UNIX propagates all currently active TASKLIB, STEPLIB and JOBLIB allocations during a fork(), spawn(), or exec() function. The RSE server starts several processes, and each client connection has a private RSE server. Because of this, the numbers can go up quickly.

Developer for System z has a default of STEPLIB=NONE coded in rsed.envvars, as described in “Customize rsed.envvars, the configuration file for RSE” on page 33. For the reasons mentioned above, it is advised not to change this directive and place the targeted data sets in LINKLIST or LPA (Link Pack Area) instead.

If you do use the STEPLIB directive, you must verify the content of rsed.envvars to see if your STEPLIB statement is the first one or not.

- If the last STEPLIB directive defined earlier in rsed.envvars equals STEPLIB=NONE
STEPLIB=first.steplib.dataset:second.steplib.dataset
- If the last STEPLIB directive defined earlier in rsed.envvars does not equal STEPLIB=NONE
STEPLIB=\$STEPLIB:first.steplib.dataset:second.steplib.dataset

Improve access to system libraries

Certain system libraries and load modules are heavily used by z/OS UNIX and application development activities. Improving access to these, like adding them to the Link Pack Area (LPA) can improve your system performance. Refer to *MVS Initialization and Tuning Reference (SA22-7592)* for more information on changing the SYS1.PARMLIB members described below.

Language Environment (LE) runtime libraries

When C programs (including the z/OS UNIX shell) are run, they frequently use routines from the Language Environment (LE) runtime library. On average, about 4 MB of the runtime library are loaded into memory for every address space running a LE-enabled program, and copied on every fork.

The CEE.SCEELPA data set contains a subset of the LE runtime routines, which are heavily used by z/OS UNIX. It is advised to add this data set to SYS1.PARMLIB(LPALSTxx) for maximum performance gain. By doing so, the modules are read from disk only once and are stored in a shared location.

Note: Add the following statement to SYS1.PARMLIB(PROGxx) if you prefer to add the load modules into dynamic LPA (Link Pack Area):

```
LPA ADD MASK(*) DSNAME(CEE.SCEELPA)
```

It is also advised to place the LE runtime libraries CEE.SCEERUN and CEE.SCEERUN2 in LINKLIST, by adding the data sets to SYS1.PARMLIB(LNKLISTxx) or SYS1.PARMLIB(PROGxx). This eliminates z/OS UNIX STEPLIB overhead and there is reduced input/output due to management by LLA and VLF, or similar products.

Note: Add the C/C++ DLL class library CBC.SCLBDLL also to LINKLIST for the same reasons.

If you decide not to put these libraries in LINKLIST, then you must set up the appropriate STEPLIB statement in rsed.envvars, as described in “Customize rsed.envvars, the configuration file for RSE” on page 33. Although this method always uses additional virtual storage, you can improve performance by defining the LE runtime libraries to LLA or a similar product. This reduces the I/O that is needed to load the modules.

Application development

On systems where application development is the primary activity, performance may also benefit if you put the linkage editor into dynamic LPA, by adding these lines to SYS1.PARMLIB(PROGxx):

```
LPA ADD MODNAME(CEEBINIT,CEEBLIBM,CEEEV003,EDCZV) DSNAME(CEE.SCEERUN)
LPA ADD MODNAME(IEFIB600,IEFXB603) DSNAME(SYS1.LINKLIB)
```

For C/C++ development, you can also add the CBC.SCCNCMP compiler data set to SYS1.PARMLIB(LPALSTxx).

The statements above are samples of possible LPA candidates, but the needs at your site may vary. Refer to *Language Environment Customization (SA22-7564)* for information on putting other LE load modules into dynamic LPA. Refer to *UNIX System Services Planning (GA22-7800)* for more information on putting C/C++ compiler load modules into dynamic LPA.

Improving performance of security checking

To improve the performance of security checking done for z/OS UNIX, define the BPX.SAFFASTPATH profile in the FACILITY class of your security software. This reduces overhead when doing z/OS UNIX security checks for a wide variety of operations. These include file access checking, IPC access checking, and process ownership checking. Refer to *UNIX System Services Planning (GA22-7800)* for more information on this profile.

Note: Users do not need to be permitted to the BPX.SAFFASTPATH profile.

Class sharing between JVMs

The IBM Java Virtual Machine (JVM) version 5 and higher allows you to share bootstrap and application classes between JVMs by storing them in a cache in shared memory. Class sharing reduces the overall virtual memory consumption when more than one JVM shares a cache. Class sharing also reduces the startup time for a JVM after the cache has been created.

The shared class cache is independent of any active JVM and persists beyond the lifetime of the JVM that created the cache. Because the shared class cache persists beyond the lifetime of any JVM, the cache is updated dynamically to reflect any modifications that might have been made to JARs or classes on the file system.

The overhead to create and populate a new cache is minimal. The JVM startup cost in time for a single JVM is typically between 0 and 5% slower compared with a system not using class sharing, depending on how many classes are loaded. JVM startup time improvement with a populated cache is typically between 10% and 40% faster compared with a system not using class sharing, depending on the operating system and the number of classes loaded. Multiple JVMs running concurrently will show greater overall startup time benefits.

Refer to the *Java SDK and Runtime Environment User Guide* to learn more about class sharing.

Enable class sharing

To enable class sharing for the RSE server, uncomment the following directive in `rsed.envvars`, as described in “(Optional) Defining extra Java startup parameters with `_RSE_*OPTS`” on page 38. The first statement defines a cache named RSE with group access and it allows the RSE server to start even if class sharing fails. The second statement is optional and it sets the cache size to 6 megabytes (system default is 16 MB).

```
_RSE_CLASS_OPTS=-Xshareclasses:name=RSE,groupAccess,nonFatal  
#_RSE_CLASS_OPTS="$_RSE_CLASS_OPTS -Xscmx6m
```

Note: As mentioned in “Cache security” on page 66, all users using the shared class must have the same primary group ID (GID). This means that the users must have the same default group defined in the security software, or that the different default groups have the same GID in their OMVS segment.

Cache size limits

The maximum theoretical shared cache size is 2 GB. The size of cache you can specify is limited by the amount of physical memory and swap space available to the system. Because the virtual address space of a process is shared between the

shared class cache and the Java heap, increasing the maximum size of the Java heap will reduce the size of the shared class cache you can create.

Cache security

Access to the shared class cache is limited by operating system permissions and Java security permissions.

By default, class caches are created with user-level security, so only the user that created the cache can access it. On z/OS UNIX, there is an option, `groupAccess`, which gives access to all users in the primary group of the user that created the cache. However, regardless of the access level used, a cache can only be destroyed by the user that created it or by a root user (UID 0).

Refer to the *Java SDK and Runtime Environment User Guide* to learn more about extra security options using a `Java SecurityManager`.

SYS1.PARMLIB(BPXPRMxx)

Some of the `SYS1.PARMLIB(BPXPRMxx)` settings affect shared classes performance. Using the wrong settings can stop shared classes from working. These settings might also have performance implications. For further information about performance implications and use of these parameters, refer to the *MVS Initialization and Tuning Reference (SA22-7592)* and *UNIX System Services Planning (GA22-7800)*. The most significant `BPXPRMxx` parameters that affect the operation of shared classes are:

- `MAXSHAREPAGES`, `IPCSHMSPAGES`, `IPCSHMMPAGES` and `IPCSHMNSEGS`

These settings affect the amount of shared memory pages available to the JVM. The shared page size for a 31-bit z/OS UNIX system service is fixed at 4 KB. Shared classes try to create a 16 MB cache by default. Therefore set `IPCSHMMPAGES` greater than 4096.

If you set a cache size using `-Xscmx`, the JVM will round up the value to the nearest megabyte. You must take this into account when setting `IPCSHMMPAGES` on your system.

- `IPCSEMNIDS` and `IPCSEMNSEMS`

These settings affect the amount of semaphores available to UNIX processes. Shared classes use IPC semaphores to communicate between the JVMs.

Disk space

The shared class cache requires disk space to store identification information about the caches that exist on the system. This information is stored in `/tmp/javasharedresources`. If the identification information directory is deleted, the JVM cannot identify the shared classes on the system and must recreate the cache.

Cache management utilities

The Java `-Xshareclasses` line command can take a number of options, some of which are cache management utilities. Three of them are shown in the sample below (`$` is the z/OS UNIX prompt). Refer to the *Java SDK and Runtime Environment User Guide* for a complete overview of supported command line options.

```
$ java -Xshareclasses:listAllCaches
Shared Cache      OS shmid      in use      Last detach time
RSE               401412        0           Mon Jun 18 17:23:16 2007
```

Could not create the Java virtual machine.


```
$ java -Xshareclasses:name=RSE,printStats
```

Current statistics for cache "RSE":

```
base address      = 0x0F300058
end address       = 0x0F8FFFF8
allocation pointer = 0x0F4D2E28
```

```
cache size        = 6291368
free bytes        = 4355696
ROMClass bytes    = 1912272
Metadata bytes    = 23400
Metadata % used   = 1%
```

```
# ROMClasses      = 475
# Classpaths      = 4
# URLs            = 0
# Tokens          = 0
# Stale classes   = 0
% Stale classes   = 0%
```

Cache is 30% full

Could not create the Java virtual machine.

```
$ java -Xshareclasses:name=RSE,destroy
JVMSHRC010I Shared Cache "RSE" is destroyed
Could not create the Java virtual machine.
```

Note: Cache utilities perform the required operation on the specified cache without starting the JVM, so the “Could not create the Java virtual machine.” message is normal.

Note: A cache can be destroyed only if all JVMs using it have shut down, and the user has sufficient permissions.

Fixed Java heap size

With a fixed-size heap, no heap expansion or contraction occurs and this can lead to significant performance gains in some situations. However, using a fixed-size heap is usually not a good idea, because it delays the start of garbage collection until the heap is full, at which point it will be a major task. It also increases the risk of fragmentation, which requires a heap compaction. Therefore, use fixed-size heaps only after proper testing or under the direction of the IBM support center. Refer to the *Java Diagnostics Guide (SC34-6650)* for more information on heap sizes and garbage collection.

By default, the initial heap size of a z/OS Java Virtual Machine (JVM) is 1 megabyte. The maximum size is 64 megabytes. The limits can be set with the `-Xms` (initial) and `-Xmx` (maximum) Java command line options.

In Developer for System z, Java command line options are defined in the `_RSE_JAVA_OPTS` directive of `rsd.envvars`, as described in “(Optional) Defining extra Java startup parameters with `_RSE_*OPTS`” on page 38.

```
#_RSE_JAVA_OPTS="$_RSE_JAVA_OPTS -Xms128m -Xmx128m"
```

Workload management

Each site has specific needs, and can customize the z/OS operating system to get the most out of the available resources to meet those needs. With workload management, you define performance goals and assign a business importance to each goal. You define the goals for work in business terms, and the system decides how much resource, such as CPU and storage, should be given to the work to meet its goal.

Developer for System z performance can be balanced by setting the correct goals for its processes. Some general guidelines are listed below.

- Assign the APPC transaction to a TSO performance group
- Assign the RSE server to a TSO performance group, or one just below it if you have significant TSO usage

Refer to *MVS Planning Workload Management (SA22-7602)* for more information on this subject.

Appendix A. Running multiple instances of Developer for System z

There are times that you want multiple instances of Developer for System z active on the same system, for example, when testing an upgrade. However, some resources like TCP/IP ports cannot be shared, so the defaults are not always applicable. Use the information in this appendix to plan the coexistence of the different instances of Developer for System z, after which you can use this configuration guide to customize them.

Although it is possible to share certain parts of Developer for System z between 2 (or more) instances, it is advised NOT to do so, unless their software levels are identical and the only changes are in configuration members. Developer for System z leaves enough customization room to make multiple instances that do not overlap and we strongly advise you to use these features.

Identical software level, different configuration files

In a limited set of circumstances, you can share all but (some of) the customizable parts. An example is providing non-SSL access for on-site usage, and SSL encoded communication for off-site usage.

Attention: The shared setup can NOT be used safely to test maintenance, a technical preview or a new release.

To set up another instance of an active Developer for System z installation, redo the customization steps for the parts that are different, using different data sets/directories/ports to avoid overlapping the current setup.

In the SSL sample mentioned above, the current RSE server customizations can be cloned, after which the cloned `ssl.properties` can be updated. The MVS customizations (JES Job Monitor etc.) can be shared between the SSL and non-SSL instances. This would result in the following actions:

1. Create a new directory to hold the SSL customized configuration members
2. Copy the active configuration members to this directory
3. Update the copied `ssl.properties` to provide the SSL related information
4. Define a new RSE daemon port number in `/etc/services`
5. Define a new RSE daemon process in `/etc/inetd.conf`, using the new directory for the `-d` parameter
6. Restart INETD to activate the new RSE daemon

All other situations

When code changes are involved (maintenance, technical previews, new release), or your changes are fairly complex, it is advised to do another installation of Developer for System z. This section describes possible points of conflict between the different installations.

The following list is a brief overview of items that must or are strongly advised to be different between the instances of Developer for System z:

- Installation libraries

- JES Job Monitor TCP/IP port, and thus its configuration file FEJJCNFG
- JES Job Monitor startup JCL
- APPC transaction name
- RSE configuration file, rsed.envvars
- RSE daemon TCP/IP port

A more detailed overview is listed below.

- Installation libraries
 1. Install each instance of Developer for System z in separate data sets and directories. Keep in mind that you can only change the z/OS UNIX path by prefixing the IBM supplied default of /usr/lpp/wd4z. A valid sample would be /service/usr/lpp/wd4z.
- Mandatory parts
 1. JES Job Monitor configuration file hlq.SFEKSAMP(FEJJCNFG) holds the TCP/IP port number of JES Job Monitor and thus cannot be shared. The member itself can be renamed (if the JCL is updated also), so you can place all customized versions of this member in the same data set if you are not doing the updates in the install data set.
 2. JES Job Monitor startup JCL hlq.SFEKSAMP(FEJJJCL) refers to FEJJCNFG and therefore cannot be shared either. After renaming the member (and the JOB card) you can place all JCL's in the same data set.
 3. The APPC transaction has a reference to hlq.SFEKPROC(FEKFRSRV), the TSO Commands server. This is software level specific, so you must create an APPC transaction per instance. Note that since version 7.1, the TSO Commands server can also be accessed through a SCLM Developer Toolkit function.
 4. The (SCLM Developer Toolkit) ISPF.conf configuration file has a reference to hlq.SFEKPROC(FEKFRSRV), the TSO Commands server. This is software level specific, so you must create an ISPF.conf file per instance. Note that is only needed if you use SCLM Developer Toolkit to access the TSO Commands server.
 5. The RSE configuration file /usr/lpp/wd4z/rse/lib/rsed.envvars holds references to the install path and the APPC transaction name, which requires it to be unique. The file name is mandatory, so you cannot keep the different copies in the same directory. Keep in mind that, since the APPC transaction name has changed, the _FEKFSCMD_TP_NAME_ variable must be defined in rsed.envvars.
 6. All mandatory configuration files and shell scripts that must reside in the same directory as rsed.envvars cannot be shared, since rsed.envvars must be in an unshared location.
 7. The REXEC and SSH TCP/IP ports can be shared without any restrictions.
 8. The RSE daemon cannot be shared since each RSE daemon owns its own TCP/IP port. A new daemon, with a different service name and port, must be created.
- Optional parts
 1. Some ELAXF* procedures have a reference to hlq.SFEKLOAD, the Developer for System z load library. See the note on JCLLIB in "Customize ELAXF*, remote build procedures" on page 21 for a possible solution on making different sets available to the users.

2. To activate two instances of the DB2 stored procedure, the following tasks must be completed. Note however that this is a non-supported, as-is description.
 - a. Copy hlq.SFEKPROC(ELAXMREX) to a differently named member, for example, ELAXMRXX
 - b. Copy sample member hlq.SFEKSAMP(ELAXMSAM) to a differently named member, for example, ELAXMWDZ
 - c. Change sample member hlq.SFEKSAMP(ELAXMJCL) to reflect these name changes, for example:


```
//SYSIN DD *
CREATE PROCEDURE SYSPROC.ELAXMRXX
  ( IN FUNCTION_REQUEST  VARCHAR(20)      CCSID EBCDIC
...
  , OUT RETURN_VALUE     VARCHAR(255)     CCSID EBCDIC )
PARAMETER STYLE GENERAL  RESULT SETS 1
LANGUAGE REXX             EXTERNAL NAME  ELAXMRXX
COLLID DSNREXCS           WLM ENVIRONMENT ELAXMWDZ
PROGRAM TYPE MAIN         MODIFIES SQL DATA
STAY RESIDENT NO          COMMIT ON RETURN NO
ASUTIME NO LIMIT          SECURITY USER;

COMMENT ON PROCEDURE SYSPROC.ELAXMRXX IS
'PLI & COBOL PROCEDURE PROCESSOR (ELAXMRXX), INTERFACE LEVEL 0.01';

GRANT EXECUTE ON PROCEDURE SYSPROC.ELAXMRXX TO PUBLIC;
//
```
 - d. Proceed with the customization as described in “(Optional) Customize ELAXM*, DB2 stored procedure members” on page 26, but with the new members.
 - e. The new WLM environment name (for example, ELAXMWDZ) must be used in the DB2 stored procedure wizard on the client.
3. bidi relies on a load library member and thus cannot be shared across releases. However, if the load module name is identical for all instances, you can share the most recent version between the instances, even across releases. Backward compatibility is not available if the load module changed name.
4. The ADM load modules are backwards compatible, and thus the most recent version can be shared across releases.
5. The ADM CRD VSAM is backwards compatible, and thus the most recent version can be shared across releases.
6. The ADM CICS resource definitions are backwards compatible, and thus the most recent version can be shared across releases.
7. All optional configuration files and shell scripts that must reside in the same directory as rsed.envvars cannot be shared, since rsed.envvars must be in an unshared location.
8. In rsed.envvars you can activate sharing Java classes between multiple RSE servers by updating the _RSE_JAVA0PTS directive. The sharing group must have a unique name (-Xshareclasses:name=RSE).
9. CARMA VSAM’s could change between software levels, thus it is not advised to share these.

Appendix B. Troubleshooting configuration problems

This appendix is provided to assist you with some common problems that you may encounter during your configuration of Developer for System z.

More information is available through the Support section of the Developer for System z website (<http://www-306.ibm.com/software/awdtools/devzseries/support/>) where you can find technotes that bring you the latest information from our support team.

In the Library section of the website you can also find the latest version of the Developer for System z documentation, including whitepapers.

Valuable information can also be found in the z/OS internet library, available at <http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/>

Location of log files

Developer for System z creates log files that can assist you and IBM support center in identifying and solving problems. The following list is an overview of log files that can be created. Next to these product specific logs, be sure to check the SYSLOG for any related messages.

MVS based logs can be located through the appropriate DD statement. z/OS UNIX based log files are located in the following directories:

- `home/.eclipse/RSE/USERID/`

Most log files are located in `home/.eclipse/RSE/USERID/`, where `home` is the home path defined in the user's OMVS segment (or the default OMVS segment if the user does not have an OMVS segment) and `USERID` is the logon user ID (uppercase).

- `daemon.log` - The log of the daemon
- `fa.log` - The log of the Fault Analyzer integration
- `fekfivpc.log` - The log of the fekfivpc IVP test
- `ffs.log` - The log of the FFS server, that executes native functions
- `ffsget.log` - The log of the file reader, that reads a sequential data set or a PDS member
- `ffsput.log` - The log of the file writer, that writes a sequential data set or a PDS member
- `lock.log` - The log of the lock manager, that locks/unlocks a sequential data set or a PDS member
- `rmt_class_loader.cache.jar` - The cache of classes loaded by the RSE remote class loader
- `rsecomm.log` - The log of the RSE server, that handles commands from the client (may contain Java exception stack trace)
- `stderr.log` - The redirected data of `stderr`, standard error output
- `stdout.log` - The redirected data of `stdout`, standard output.

- `/tmp/`

`/tmp/` holds log files created before the user ID has been verified.

- `rsedaemon.log` - The log of the daemon before login

JES Job Monitor logging

- **SYSOUT DD**

Logging of normal operations. The default value in the sample JCL h1q.SFEKSAMP(FEJJJCL) is SYSOUT=*.

- **SYSPRINT DD**

Trace logging. The default value in the sample JCL h1q.SFEKSAMP(FEJJJCL) is SYSOUT=*. Tracing is activated with the -TV parameter, see “Customize the JES Job Monitor startup JCL” on page 18 for more details.

APPC transaction (TSO Commands service) logging

- **SYSPRINT DD**

When the APPC administration utility adds and modifies a transaction program (TP) profile, it checks the TP profile and its JCL for syntax errors. Output from this phase consists of TP profile syntax error messages, utility processing messages, and JCL conversion statements. Logging for messages from this phase is controlled by the SYSPRINT DD statement for the ATBSEDFMU utility. The default value in sample JCL h1q.SFEKSAMP(FEKAPPCC) is SYSOUT=*. See *MVS Planning APPC/MVS Management (SA22-7599)* for more details.

- **&SYSUID.FEKFRSRV.&TPDATE.&TPTIME.LOG**

When a TP executes, the TP runtime messages, such as allocation and termination messages, go to a log named by the MESSAGE_DATA_SET keyword in its TP profile. The default value in sample JCL h1q.SFEKSAMP(FEKAPPCC) is &SYSUID.FEKFRSRV.&TPDATE.&TPTIME.LOG See *MVS Planning APPC/MVS Management (SA22-7599)* for more details.

Note: Depending on your APPC transaction definitions and site defaults, this log file might not appear unless the KEEP_MESSAGE_LOG(ALWAYS) keyword is added to the transaction definitions. Refer to *MVS Planning APPC/MVS Management (SA22-7599)* for more information on this.

RSE logging

- **home/.eclipse/RSE/USERID/**

There are several log files created by the components related to RSE, most of them located in home/.eclipse/RSE/USERID/, where home is the home path defined in the user's OMVS segment (or the default OMVS segment if the user does not have an OMVS segment) and USERID is the logon user ID (uppercase).

- daemon.log - The log of the daemon
- ffs.log - The log of the FFS server, that executes native MVS functions
- ffsget.log - The log of the file reader, that reads a sequential data set or a PDS member
- ffsput.log - The log of the file writer, that writes a sequential data set or a PDS member
- lock.log - The log of the lock manager, that locks/unlocks a sequential data set or a PDS member
- rmt_class_loader.cache.jar - The cache of classes loaded by the RSE remote class loader
- rsecomm.log - The log of the RSE server, that handles commands from the client (may contain Java exception stack trace)
- stderr.log - The redirected data of stderr, standard error output.
- stdout.log - The redirected data of stdout, standard output.

The amount of data written to `ffs*.log`, `lock.log` and `rsecomm.log` is controlled by setting `debug_level` in `rsecomm.properties`. See “(Optional) Customize `rsecomm.properties`, RSE trace configuration” on page 49 for more details.

- **`/tmp/rsedaemon.log`**

This file contains a log of the daemon before login.

fekfivpc IVP test logging

- **`home/.eclipse/RSE/USERID/fekfivpc.log`**

Output of the `fekfivpc -file` command (SCLM Developer Toolkit related IVP test), where `home` is the home path defined in the user’s OMVS segment (or the default OMVS segment if the user does not have an OMVS segment) and `USERID` is the logon user ID (uppercase).

Refer to “TSO Commands service connection (using `SCLMDT`)” on page 47 for more details.

Fault Analyzer Integration logging

- **`home/.eclipse/RSE/USERID/fa.log`**

Fault Analyzer integration logging, where `home` is the home path defined in the user’s OMVS segment (or the default OMVS segment if the user does not have an OMVS segment) and `USERID` is the logon user ID (uppercase).

File Manager Integration logging

- **File Manager Integration server job**

When opening a connection with File Manager, a server job will be started, with the user’s user ID as owner. Its name is `FEKport`, where `port` is the TCP/IP port used.

- **SYSPRINT DD**

The SYSPRINT of the server job holds the FMI server logging.

- **`home/.eclipse/RSE/USERID/rsecomm.log`**

Communication logging of FMI, where `home` is the home path defined in the user’s OMVS segment (or the default OMVS segment if the user does not have an OMVS segment) and `USERID` is the logon user ID (uppercase).

The amount of data written to this file is controlled by setting `debug_level` in `rsecomm.properties`. See “(Optional) Customize `rsecomm.properties`, RSE trace configuration” on page 49 for more details.

CARMA logging

- **CARMA server job**

When opening a connection with CARMA, `hlq.SCRACLST(SCRASUBMT)` will start a server job (with the user’s user ID as owner) named `CRAport`, where `port` is the TCP/IP port used.

- **CARMALOG DD**

If DD statement `CARMALOG` is specified in `hlq.SCRACLST(SCRASUBMT)`, CARMA logging is redirected to this DD statement in the server job, otherwise it goes to SYSPRINT.

The amount of logging is controlled by setting `Trace Level` on the client. See Chapter 5, “(Optional) Activating IBM Common Access Repository Manager (CARMA),” on page 53 for more details on setting `Trace Level`.

- **SYSPRINT DD**

The SYSPRINT of the server job holds the CARMA logging, if DD statement CARMALOG is not defined.

- **home/.eclipse/RSE/USERID/rsecomm.log**

Communication logging of CARMA, where home is the home path defined in the user's OMVS segment (or the default OMVS segment if the user does not have and OMVS segment) and USERID is the logon user ID (uppercase).

The amount of data written to this file is controlled by setting `debug_level` in `rsecomm.properties`. See "(Optional) Customize `rsecomm.properties`, RSE trace configuration" on page 49 for more details.

Dump files

When a product abnormally terminates, a storage dump is created to assist in problem determination. The availability and location of these dumps depends heavily on site specific settings. So it could be that they are not created, or created in different locations than mentioned below.

MVS dumps

When the program is running in MVS, check the system dump files and check your JCL for the following DD statements (depending on the product):

- SYSABEND
- SYSMDUMP
- SYSUDUMP
- CEEDUMP
- SYSPRINT
- SYSOUT

Refer to the *MVS JCL Reference* (SA22-7597) and the *Language Environment Debugging Guide* (GA22-7560) for more information on these DD statements.

Java dumps

In z/OS UNIX, Developer for System z dumps are controlled by the Java Virtual Machine (JVM).

The JVM creates a set of dump agents by default during its initialization (SYSTDUMP and JAVADUMP). You can override this set of dump agents using the `JAVA_DUMP_OPTS` environment variable and further override the set by the use of `-Xdump` on the command line. JVM command line options are defined in the `_RSE_JAVA_OPTS` directive of `rsed.envvars`. Do not change any of the dump settings unless directed by the IBM support center.

Note: The `-Xdump:what` option on the command line can be used for determining which dump agents exist at the completion of startup.

The types of dump that can be produced are:

SYSTDUMP

Java Transaction dump. An unformatted storage dump generated by z/OS.

The dump is written to a sequential MVS data set, using a default name of the form `&userid.JVM.TDUMP.&jobname.D&date.T&time`, or as determined by the setting of the `JAVA_DUMP_TDUMP_PATTERN` environment variable. If you do not want transaction dumps to be created, add environment variable `IBM_JAVA_ZOS_TDUMP=NO` to `rsed.envvars`.

CEEDUMP

Language Environment (LE) dump. A formatted summary system dump that shows stack traces for each thread that is in the JVM process, together with register information and a short dump of storage for each register.

The dump is written to a z/OS UNIX file named `CEEDUMP.yyyymmdd.hhmmss.pid`, where `yyymmdd` equals the current date, `hhmmss` the current time and `pid` the current process ID. The possible locations of this file are described in “z/OS UNIX dump locations.”

HEAPDUMP

A formatted dump (a list) of the objects that are on the Java heap.

The dump is written to a z/OS UNIX file named `HEAPDUMP.yyyymmdd.hhmmss.pid.TXT`, where `yyymmdd` equals the current date, `hhmmss` the current time and `pid` the current process ID. The possible locations of this file are described in “z/OS UNIX dump locations.”

JAVADUMP

A formatted analysis of the JVM. It contains diagnostic information related to the JVM and the Java application, like the application environment, threads, native stack, locks, and memory.

The dump is written to a z/OS UNIX file named `JAVADUMP.yyyymmdd.hhmmss.pid.TXT`, where `yyymmdd` equals the current date, `hhmmss` the current time and `pid` the current process ID. The possible locations of this file are described in “z/OS UNIX dump locations.”

Refer to the *Java Diagnostics Guide (SC34-6650)* for more information on JVM dumps, and the *Language Environment Debugging Guide (GA22-7560)* for LE specific information.

z/OS UNIX dump locations

The JVM checks each of the following locations for existence and write-permission, and stores the CEEDUMP, HEAPDUMP and JAVADUMP files in the first one available. Note that you must have enough free disk space for the dump file to be written correctly.

1. The directory in environment variable `_CEE_DMPTARG`, if found. This variable is set in `rsed.envvars` as `home/.eclipse/RSE/USERID`, where `home` is the home path defined in the user's OMVS segment (or the default OMVS segment if the user does not have an OMVS segment) and `USERID` is the logon user ID (uppercase).
2. The current working directory, if the directory is not the root directory (`/`), and the directory is writable.
3. The directory in environment variable `TMPDIR` (an environment variable that indicates the location of a temporary directory if it is not `/tmp`), if found.
4. The `/tmp` directory.
5. If the dump cannot be stored in any of the above, it is put to `stderr`.

Program Control authorization for RSE programs

Remote Systems Explorer (RSE) is the Developer for System z component that provides core services like connecting the client to the host. It must run program controlled in order to perform tasks like switching to the user ID of the client.

The program control bit is set during SMP/E install where needed.

This result can be verified with the **ls -lE fekf*** command, which gives an output like the following sample (\$ is the z/OS UNIX prompt):

```
$ ls -lE /usr/lpp/wd4z/rse/lib/fekf*
-rwxr-xr-x -ps- 2 user group 94208 Jun 12 16:21 /usr/lpp/wd4z/rse/lib/fekfdir.dll
-rwxr-xr-x -ps- 2 user group 143360 Jun 12 16:21 /usr/lpp/wd4z/rse/lib/fekfdivp
-rwxr-xr-x --s- 2 user group 480 Jun 12 16:21 /usr/lpp/wd4z/rse/lib/fekfivpa
-rwxr-xr-x --s- 2 user group 342 Jun 12 16:21 /usr/lpp/wd4z/rse/lib/fekfivpc
-rwxr-xr-x --s- 2 user group 445 Jun 12 16:21 /usr/lpp/wd4z/rse/lib/fekfivpd
-rwxr-xr-x --s- 2 user group 1491 Jun 12 16:21 /usr/lpp/wd4z/rse/lib/fekfivpj
-rwxr-xr-x --s- 2 user group 883 Jun 12 16:21 /usr/lpp/wd4z/rse/lib/fekfivpr
-rwxr-xr-x --s- 2 user group 307 Jun 12 16:21 /usr/lpp/wd4z/rse/lib/fekfivps
-rwxr-xr-x -ps- 2 user group 139264 Jun 12 16:21 /usr/lpp/wd4z/rse/lib/fekflock
-rwxr-xr-x -ps- 2 user group 196608 Jun 12 16:21 /usr/lpp/wd4z/rse/lib/fekfrsed
-rwxr-xr-x --s- 2 user group 42443 Jun 12 16:21 /usr/lpp/wd4z/rse/lib/fekfscmd
```

Note: fekfivp* and fekfscmd do not require the +p attribute.

Issue the following commands if the program control bit needs to be set manually, assuming that the default install directory (/usr/lpp/wd4z/rse/lib/) is used:

1. cd /usr/lpp/wd4z/rse/lib
2. extattr +p fekfdir.dll fekfivp fekflock fekfrsed

Note: To be able to use the **extattr +p** command, you must have at least READ access to the BPX.FILEATTR.PROGCTL profile in the FACILITY class of your security software. For more information, see *UNIX System Services Planning (GA22-7800)*.

Reserved TCP/IP ports

With the **NETSTAT** command (TSO or z/OS UNIX) you can get an overview of the ports currently in use. The output of this command will look similar to one of the examples below. The ports used are the last number (behind the '..') in the "Local Socket" column/row. Since these ports are already in use, they cannot be used for the Developer for System z configuration.

IPv4

MVS TCP/IP NETSTAT CS V1R7			TCP/IP Name: TCP/IP		16:36:42
User Id	Conn	Local Socket	Foreign Socket		State
-----	----	-----	-----		----
BPX0INIT	00000014	0.0.0.0..10007	0.0.0.0..0		Listen
INETD4	0000004D	0.0.0.0..512	0.0.0.0..0		Listen
INETD4	0000004B	0.0.0.0..4035	0.0.0.0..0		Listen
JMON	00000038	0.0.0.0..6715	0.0.0.0..0		Listen

IPv6

MVS TCP/IP NETSTAT CS V1R7			TCP/IP Name: TCP/IP		12:46:25
User Id	Conn	State			
-----	----	-----			
BPX0INIT	00000018	Listen			
	Local Socket:	0.0.0.0..10007			
	Foreign Socket:	0.0.0.0..0			
INETD4	00000046	Listen			
	Local Socket:	0.0.0.0..512			
	Foreign Socket:	0.0.0.0..0			
INETD4	0000004B	Listen			
	Local Socket:	0.0.0.0..4035			

```

Foreign Socket: 0.0.0.0..0
JMON      00000037 Listen
Local Socket:  0.0.0.0..6715
Foreign Socket: 0.0.0.0..0

```

Another limitation that can exist is reserved TCP/IP ports. There are 2 common places to reserve TCP/IP ports:

- **PROFILE.TCPIP**

This is the data set referred to by the PROFILE DD statement of the TCP/IP started task, often named SYS1.TCPPARMS(TCPPROF)

- PORT: Reserves a port for specified job names.
- PORTRANGE: Reserves a range of ports for specified job names.

Refer to *Communications Server IP Configuration Guide (SC31-8775)* for more information on these statements.

- **SYS1.PARMLIB(BPXPRMxx)**

- INADDRANYPORT: Specifies the starting port number for the range of port numbers that the system reserves for use with PORT 0, INADDR_ANY binds. This value is only needed for CINET.
- INADDRANYCOUNT: Specifies the number of ports that the system reserves, starting with the port number specified in the INADDRANYPORT parameter. This value is only needed for CINET.

Refer to *UNIX System Services Planning (GA22-7800)* and *MVS Initialization and Tuning Reference (SA22-7592)* for more information on these statements.

These reserved ports can be listed with the **NETSTAT PORTL** command (TSO or z/OS UNIX), which creates an output like in the example below:

```

MVS TCP/IP NETSTAT CS V1R7      TCPIP Name: TCPIP      17:08:32
Port# Prot User   Flags   Range      IP Address
-----
00007 TCP  MISCSERV DA
00009 TCP  MISCSERV DA
00019 TCP  MISCSERV DA
00020 TCP  OMVS     D
00021 TCP  FTPD1    DA
00025 TCP  SMTP     DA
00053 TCP  NAMESRV  DA
00080 TCP  OMVS     DA
03500 TCP  OMVS     DAR      03500-03519
03501 TCP  OMVS     DAR      03500-03519

```

Refer to *Communications Server IP System Administrator's Commands (SC31-8781)* for more information on the **NETSTAT** command.

Note: The **NETSTAT** command only shows the information defined in PROFILE.TCPIP, which should overlap the BPXPRMxx definitions. In case of doubt or problems, check the BPXPRMxx parmlib member to verify the ports being reserved here.

Note: Read “PROFILE.TCPIP port definitions” on page 95 if you have problems with INETD binding to reserved ports.

Address Space size

The RSE server, which is a Java UNIX process, requires a large region size to perform its functions. Therefore it is important to set large storage limits for OMVS address spaces.

INETD requirements

An RSE server is started by INETD when a client connects to the RSE daemon port or when the client issues the startup script using REXEC/SSH. This is done using INETD's storage limits, so these must be set large enough.

- If the UNIX server process INETD is started by JCL using BPXBATCH, the region size must be 0.
- If INETD is started from a TSO/OMVS session, the TSO region size must be 2096128 or larger.

Limitations set in SYS1.PARMLIB(BPXPRMxx)

Set MAXASSIZE in SYS1.PARMLIB(BPXPRMxx), which defines the default OMVS address space (process) region size, to 2147483647 or larger.

This value can be checked and set dynamically (until the next IPL) with the following console commands, as described in *MVS System Commands (SA22-7627)*:

1. DISPLAY OMVS,0
2. SETOMVS MAXASSIZE=2147483647

Limitations stored in the security profile

Check ASSIZEMAX in the client's user ID OMVS segment, and set it to 2147483647 or, preferably, to NONE to use the SYS1.PARMLIB(BPXPRMxx) value.

Using RACF, this value can be checked and set with the following TSO commands, as described in *Security Server RACF Command Language Reference (SA22-7687)*:

1. LISTUSER userid NORACF OMVS
2. ALTUSER userid OMVS(NOASSIZEMAX)

Limitations enforced by system exits

Make sure you're not allowing system exits IEFUSI or IEALIMIT to control OMVS address space region sizes. A possible way to accomplish this is by coding SUBSYS(OMVS,NOEXITS) in SYS1.PARMLIB(SMFPRMxx).

SYS1.PARMLIB(SMFPRMxx) values can be checked and activated with the following console commands, as described in *MVS System Commands (SA22-7627)*:

1. DISPLAY SMF,0
2. SET SMF=xx

Error feedback tracing

The following procedure allows gathering of information needed to diagnosis error feedback problems with remote build procedures. This tracing will cause performance degradation and should only be done under the direction of the IBM support center. All references to hlq in this section refer to the high level qualifier used during installation of Developer for System z. The installation default is FEK, but this might not apply to your site.

1. Make a backup copy of your active ELAXFC0C compile procedure. This procedure is default shipped in data set hlq.SFEKSAMP, but could have been copied to a different location, like SYS1.PROCLIB, as described in "Customize ELAXF*, remote build procedures" on page 21.
2. Change the active ELAXFC0C procedure to include the 'MAXTRACE' string on the EXIT(ADEXIT(ELAXMGUX)) compile option.

```
//COBOL EXEC PGM=IGYCRCTL,REGION=2048K,
//*      PARM=('EXIT(ADEXIT(ELAXMGUX))'),
//      PARM=('EXIT(ADEXIT('MAXTRACE',ELAXMGUX))',
//      'ADATA',
//      'LIB',
//      'TEST(NONE,SYM,SEP)',
//      'LIST',
//      'FLAG(I,I) &CICS &DB2 &COMP)
```

Note: You have to double the apostrophes around MAXTRACE. The option is now: EXIT(ADEXIT('MAXTRACE',ELAXMGUX))

3. Perform a Remote Syntax Check on a COBOL program. Developer for System z ships a sample COBOL program in hlq.SFEKSAMP(ADNSMSGH).
4. The SYSOUT part of the JES output will start by listing the names of the data sets for SIDEFILE1, SIDEFILE2, SIDEFILE3 and SIDEFILE4.

```
ABOUT TOO OPEN SIDEFILE1 - NAME = 'uid.DT021207.TT110823.M0000045.C0000000'
SUCCESSFUL OPEN SIDEFILE1 - NAME = 'uid.DT021207.TT110823.M0000045.C0000000'
ABOUT TOO OPEN SIDEFILE2 - NAME = 'uid.DT021207.TT110823.M0000111.C0000001'
SUCCESSFUL OPEN SIDEFILE2 - NAME = 'uid.DT021207.TT110823.M0000111.C0000001'
ABOUT TOO OPEN SIDEFILE3 - NAME = 'uid.DT021207.TT110823.M0000174.C0000002'
SUCCESSFUL OPEN SIDEFILE3 - NAME = 'uid.DT021207.TT110823.M0000174.C0000002'
ABOUT TOO OPEN SIDEFILE4 - NAME = 'uid.DT021207.TT110823.M0000236.C0000003'
SUCCESSFUL OPEN SIDEFILE4 - NAME = 'uid.DT021207.TT110823.M0000236.C0000003'
```

Note: Depending on your settings, SIDEFILE1 and SIDEFILE2 may be pointing to a DD statement (SUCCESSFUL OPEN SIDEFILE1 - NAME = DD:WSEDSF1). Refer to the JESJCL part of the output (which is located before the SYSOUT part) to get the actual data set name.

```
22 //COBOL.WSEDSF1 DD DISP=MOD,
// DSN=uid.ERRCOB.member.SF1.Z682746.XML
23 //COBOL.WSEDSF2 DD DISP=MOD,
// DSN=uid.ERRCOB.member.SF1.Z682747.XML
```

5. Copy these four data sets to your PC, for example by creating a local COBOL project in Developer for System z and adding the SIDEFILE1->4 data sets.
6. Copy the complete JES job log to your PC, for example by opening the job output in Developer for System z and saving it to the local project by selecting **File > Save As ...**.
7. Restore procedure ELAXFCOC to the original state, either by undoing the change (remove the "MAXTRACE", string in the compile options) or restoring the backup.
8. Send the collected files (SIDEFILE1->4 and job log) to the IBM support center.

APPC transaction and TSO Commands service

If you cannot use the APPC version of the TSO Commands service, there are two areas where problems may arise: starting the APPC server transaction and connecting to RSE.

- If you do not see the messages about setting up APPC, check the system log for RACF messages (message id ICHxxxxx) 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 hlq.SFEKPROC data set.
 - TCP/IP is not active, has a wrong DNS name attached or the system is unreachable (not pingable) due to network problems, a bad IP address or other causes.

- 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. Check the transaction error log (userid.FEKFRSRV.&TPDATE.&TPTIME.LOG). 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 must include hlq.SFEKPROC).
 - The server could not open or access the message (log) data set referred to by MESSAGE_DATA_SET.
 - There are not enough APPC scheduler initiators available.
 - APPC or ASCH address spaces are not active.
 - The class used (default 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 a personal OMVS segment, or there is a definition error in either.
 - The default group of the default OMVS segment or the default group of the user does not have a GID number.

The REXX provided in “(Optional) Define an APPC transaction for the TSO Commands service” on page 23 can help with solving APPC problems since it gives you the possibility to manage APPC interactively through ISPF panels. Be aware however that you can deactivate the transaction with this tool; the transaction is still there but won’t accept any connections.

The following list is a selection of technotes currently available on the support website (<http://www-306.ibm.com/software/awdtools/devzseries/support/>). Please refer to the support website for additional information:

- APPC verification fails with Return code 2016 - EHOSTNOTFOUND
- APPC verification fails with Return code 1004 - EIBMIUCVERR
- APPC verification fails with Return code 9 - TP Name not recognized
- APPC verification fails with Return code 10 - TP not avail no retry
- APPC verification fails with Return code 19 - Parameter Error
- APPC verification fails with Return code 20 - Product specific error
- APPC verification fails with Return code 26 - Resource failure
- CEE3501S: The module IOSTREAM was not found
- Server Failed to Start: EDC5129I No such file or directory
- exec/tcp: bind: EDC5111I Permission denied, rsn=744C7246
- No response from server, with either one of these messages
 - IEA995I SYMPTOM DUMP OUTPUT 473 USER COMPLETION CODE=4093 REASON CODE=0000001C (in SDSF LOG)
 - CEE3512S An HFS load of module libicudata32.0.dll failed. The system return code was 0000000157; the reason code was 0BDF019B. (in CEEDUMP)
 - Get Space failed (in client .log)
- Command C_CONNECT is not available
- “FFS server initialization failed” error message when connecting to the host
- “EDC5139I Operation not permitted” when connecting to the host
- “RSEG1056U FFS server initialization failed” when opening an MVS file

Note: This list is not definitive, please check the support website for additional technotes.

Miscellaneous information

System limits

SYS1.PARMLIB(BPXPRMxx) defines many z/OS UNIX related limitations, which might be reached when several Developer for System z clients are active. Most BPXPRMxx values can be changed dynamically with the **SETOMVS** and **SET OMVS** console commands.

Connection refused

Each RSE connection starts several processes which are permanently active. New connections can be refused due to the limit set in SYS1.PARMLIB(BPXPRMxx) on the amount of processes, especially when users share the same UID (like when using the default OMVS segment).

- The limit per UID is set by the MAXPROCUSER keyword and has a default value of 25.
- The system-wide limit is set by the MAXPROCSYS keyword and has a default value of 200.

Another source of refused connections is the limit on the amount of active z/OS address spaces and z/OS UNIX users.

- The maximum amount of Address Space ID's (ASID) is defined in SYS1.PARMLIB(IEASYSxx) with the MAXUSER keyword, and has a default value of 255.
- The maximum amount of z/OS UNIX user IDs (UID) is defined in SYS1.PARMLIB(BPXPRMxx) with the MAXUIDS keyword, and has a default value of 200.

Known Issues

Opening MVS data sets fails

Reading and writing a MVS data set requires the use of a socket physical file system domain. If the file system is not properly defined or it has not enough sockets, the lock manager (FFS) might fail read/write requests. The ffs*.log files will show messages like the following:

- Error 127 getting socket pair - setting port to 0.
- Unable to create socket in the UNIX domain. Error is: "The address family is not supported"

Verify that the SYS1.PARMLIB(BPXPRMxx) member contains the following statements:

```
FILESYSTYPE TYPE(UDS) ENTRYPPOINT(BPXTUINT)
NETWORK DOMAINNAME(AF_UNIX)
          DOMAINNUMBER(1)
          MAXSOCKETS(200)
          TYPE(UDS)
```

DVIPA binds fail

When using dynamic VIPA (Virtual IP address) across multiple TCPIP stacks, sysplex-wide coordination of assignment of ephemeral ports is required so that the 4-tuple (combination of source and destination IP addresses and ports) for each connection remains unique. This can be done by adding the optional SYSPLXEXPORTS

parameter to the first VIPADISTRIBUTE statement, as described in the *Communications Server IP Configuration Guide (SC31-8775)*.

When you use this technique, ensure that the EZBEP0RT coupling facility structure (containing sysplex port assignment information) has been defined. Refer to the *Communications Server SNA Network Implementation Guide (SC31-8777)* for information on how to do this.

Host Connect Emulator

- Host Connect Emulator uses TN3270 telnet and not the RSE server to connect to the host.
- When using secure telnet (SSL) and you are working with certificates that are not signed by a well-known CA, every client must add the CA certificate to their Host Connect Emulator list of trusted CA's.
- The NOSNAEXT option of TCP/IP's TELNETPARMS might be necessary to disable the SNA functional extensions. If NOSNAEXT is specified, the TN3270 telnet server does not negotiate for contention resolution and SNA sense functions.

Contacting IBM support

If you are still experiencing difficulty after reading this manual and checking the support website (<http://www-306.ibm.com/software/awdtools/devzseries/support/>), and assistance is required from IBM support, please gather the following information and open a problem record with IBM:

- Your current software level, which is documented in:
 - hlq.SFEKJCL(FEK#LVL#) for Developer for System z. The default value for hlq is FEK.
 - hlq.SCRAJCL(CRA#LVL#) for CARMA. The default value for hlq is CRA.

The following information is useful for solving connection problems

- The output of the following console command(s):
 - D OMVS,0
- The output of the following TSO command(s):
 - HOMETEST
 - LISTUSER userid NORACF OMVS
 - LISTGROUP group NORACF OMVS
- File rsed.envvars, default located in /usr/lpp/wd4z/rse/lib/, but advised to be customized in /etc/wd4z/
- File /etc/inetd.conf
- All *.log files located in home/.eclipse/RSE/USERID/, where home is the home path defined in the user's OMVS segment (or the default OMVS segment if the user does not have an OMVS segment) and USERID is the logon user ID (uppercase).
- Related SYSLOG messages
- Complete output of the related IVP test(s), as described in "Verify RSE server set up" on page 43.
 - netstat : "Port availability" on page 44
 - fekfivps : "REXEC/SSH shell script" on page 46
 - fekfivpr : "REXEC connection" on page 45
 - fekfivpd : "RSE daemon connection" on page 46.

- fekfivpj : “JES Job Monitor connection” on page 47.
- fekfivpc : “TSO Commands service connection (using SCLMDT)” on page 47
- fekfivpa : “TSO Commands service connection (using APPC)” on page 48

If you use SCLM Developer Toolkit for the TSO Commands service (default method)

- Complete output of the SCLMDT IVP test (fekfivpc), as described in “TSO Commands service connection (using SCLMDT)” on page 47
- File /etc/SCLMDT/CONFIG/ISPF.conf

If you use APPC for the TSO Commands service

- Complete output of the APPC IVP test (fekfivpa), as described in “TSO Commands service connection (using APPC)” on page 48
- The JCL that defines the APPC server transaction, which is hlq.SFEKSAMP(FEKAPPCC) if you used the sample job
- Member SYS1.PARMLIB(APPCPMxx)
- Member SYS1.PARMLIB(ASCHPMxx)
- Data set userid.FEKFSERV.&TPDATE.&TPTIME.LOG
- The output of the following console commands:
 - D APPC,TP,ALL
 - D APPC,LU,ALL
 - D ASCH,ALL

Provide all information that seems relevant for functional problems, like

- Job logging
- Related SYSLOG messages
- Log files located in home/.eclipse/RSE/USERID/, where home is the home path defined in the user’s OMVS segment (or the default OMVS segment if the user does not have an OMVS segment) and USERID is the logon user ID (uppercase).

Appendix C. Setting up TCP/IP

This appendix is provided to assist you with some common problems that you may encounter when setting up TCP/IP, or during checking and/or modifying an existing setup.

Refer to *Communications Server IP Configuration Guide (SC31-8775)* and *Communications Server IP Configuration Reference (SC31-8776)* for additional information on TCP/IP configuration.

Hostname dependency

Developer for System z is dependant upon TCP/IP having the correct hostname when it is initialized. This implies that the different TCP/IP and Resolver configuration files must be set up correctly.

You can test your TCP/IP configuration with the TSO command **HOMETEST**. Refer to *Communications Server IP System Administrator's Commands (SC31-8781)* for more information on the **HOMETEST** command.

Example output of the **HOMETEST** command:

Running IBM MVS TCP/IP CS V1R7 TCP/IP Configuration Tester

The FTP configuration parameter file used will be "SYS1.TCPPARMS(FTPDATA)"

TCP Host Name is: CDFMVS08

Using Name Server to Resolve CDFMVS08

The following IP addresses correspond to TCP Host Name: CDFMVS08
9.42.112.75

The following IP addresses are the HOME IP addresses defined in PROFILE.TCPIP:
9.42.112.75
127.0.0.1

All IP addresses for CDFMVS08 are in the HOME list!

Hometest was successful – all Tests Passed!

Understanding resolvers

The resolver acts on behalf of programs as a client that accesses name servers for name-to-address or address-to-name resolution. To resolve the query for the requesting program, the resolver can access available name servers, use local definitions (for example, `/etc/resolv.conf`, `/etc/hosts`, `/etc/ipnodes`, `HOSTS.SITEINFO`, `HOSTS.ADDRINFO` or `ETC.IPNODES`), or use a combination of both.

When the resolver address space starts, it reads an optional resolver setup data set pointed to by the SETUP DD card in the resolver JCL procedure. If the setup information is not provided, the resolver uses the applicable native MVS or z/OS UNIX search order without any `GLOBALTCPIPDATA`, `DEFAULTTCPIPDATA`, `GLOBALIPNODES`, `DEFAULTIPNODES` or `COMMONSEARCH` information.

Note: If no resolver JCL procedure is present in the system proclib, the address space will start using system defaults (no SETUP DD).

Understanding search orders of configuration information

It is important to understand the search order for configuration files used by TCP/IP functions, and when you can override the default search order with environment variables, JCL, or other variables you provide. This knowledge allows you to accommodate your local data set and HFS file naming standards, and it is helpful to know the configuration data set or HFS file in use when diagnosing problems.

Another important point to note is that when a search order is applied for any configuration file, the search ends with the first file found. Therefore, unexpected results are possible if you place configuration information in a file that never gets found, either because other files exist earlier in the search order, or because the file is not included in the search order chosen by the application.

When searching for configuration files, you can explicitly tell TCP/IP where most configuration files are by using DD statements in the JCL procedures or by setting environment variables. Otherwise, you can let TCP/IP dynamically determine the location of the configuration files, based on search orders documented in *Communications Server IP Configuration Guide (SC31-8775)*.

The TCP/IP stack's configuration component uses TCPIP.DATA during TCP/IP stack initialization to determine the stack's HOSTNAME. To get its value, the z/OS UNIX environment search order is used.

Note: Use the trace resolver facility to determine what TCPIP.DATA values are being used by the resolver and where they were read from. For information on dynamically starting the trace, refer to *Communications Server IP Diagnosis Guide (GC31-8782)*. Once the trace is active, issue a TSO **NETSTAT HOME** command and a z/OS UNIX shell **netstat -h** command to display the values. Issuing a PING of a host name from TSO and from the z/OS UNIX shell also shows activity to any DNS servers that might be configured.

Search orders used in the z/OS UNIX environment

The particular file or table that is searched for can be either an MVS data set or an HFS file, depending on the resolver configuration settings and the presence of given files on the system.

Base resolver configuration files:

The base resolver configuration file contains TCPIP.DATA statements. In addition to resolver directives, it is referenced to determine, among other things, the data set prefix (DATASETPREFIX statement's value) to be used when trying to access some of the configuration files specified in this section.

The search order used to access the base resolver configuration file is as follows:

1. **GLOBALTCPIPDATA**

If defined, the resolver GLOBALTCPIPDATA setup statement value is used (see also "Understanding resolvers" on page 87). The search continues for an additional configuration file. The search ends with the next file found.

2. The value of the environment variable **RESOLVER_CONFIG**

The value of the environment variable is used. This search will fail if the file does not exist or is allocated exclusively elsewhere.

3. **/etc/resolv.conf**

4. **//SYSTCPD DD card**

The data set allocated to the DD name SYSTCPD is used. In the z/OS UNIX environment, a child process does not have access to the SYSTCPD DD. This is because the SYSTCPD allocation is not inherited from the parent process over the fork() or exec function calls.

5. **userid.TCPIP.DATA**

userid is the user ID that is associated with the current security environment (address space or task/thread).

6. **jobname.TCPIP.DATA**

jobname is the name specified on the JOB JCL statement for batch jobs or the procedure name for a started procedure.

7. **SYS1.TCPPARMS(TCPDATA)**

8. **DEFAULTTCPIPDATA**

If defined, the resolver DEFAULTTCPIPDATA setup statement value is used (see also “Understanding resolvers” on page 87).

9. **TCPIP.TCPIP.DATA**

Translate tables:

The translate tables (EBCDIC-to-ASCII and ASCII-to-EBCDIC) are referenced to determine the translate data sets to be used. The search order used to access this configuration file is as follows. The search order ends at the first file found:

1. The value of the environment variable **X_XLATE** The value of the environment variable is the name of the translate table produced by the TSO CONVXLAT command.
2. **userid.STANDARD.TCPXLBIN**
userid is the user ID that is associated with the current security environment (address space or task/thread).
3. **jobname.STANDARD.TCPXLBIN**
jobname is the name specified on the JOB JCL statement for batch jobs or the procedure name for a started procedure.
4. **hlq.STANDARD.TCPXLBIN**
hlq represents the value of the DATASETPREFIX statement specified in the base resolver configuration file (if found); otherwise, hlq is TCPIP by default.
5. If no table is found, the resolver uses a hard coded default table, identical to the table listed in data set member SEZATCPX(STANDARD).

Local host tables:

By default, resolver first attempts to use any configured domain name servers for resolution requests. If the resolution request cannot be satisfied, local host tables are used. Resolver behavior is controlled by TCPIP.DATA statements.

The TCPIP.DATA resolver statements define if and how domain name servers are to be used. The LOOKUP TCPIP.DATA statement can also be used to control how domain name servers and local host tables are used. For more information on TCPIP.DATA statements, refer to *Communications Server IP Configuration Reference (SC31-8776)*.

The resolver uses the Ipv4-unique search order for sitename information unconditionally for getnetbyname API calls. The Ipv4-unique search order for sitename information is as follows. The search ends at the first file found:

1. The value of the environment variable **X_SITE**

The value of the environment variable is the name of the HOSTS.SITEINFO information file created by the TSO **MAKESITE** command.

2. The value of the environment variable **X_ADDR**

The value of the environment variable is the name of the HOSTS.ADDRINFO information file created by the TSO **MAKESITE** command.

3. **/etc/hosts**

4. **userid.HOSTS.SITEINFO**

userid is the user ID that is associated with the current security environment (address space or task/thread).

5. **jobname.HOSTS.SITEINFO**

jobname is the name specified on the JOB JCL statement for batch jobs or the procedure name for a started procedure.

6. **hlq.HOSTS.SITEINFO**

hlq represents the value of the DATASETPREFIX statement specified in the base resolver configuration file (if found); otherwise, hlq is TCPIP by default.

Applying this to Developer for System z

As stated before, Developer for System z is dependant upon TCP/IP having the correct hostname when it is initialized. This implies that the different TCP/IP and Resolver configuration files must be set up correctly.

In the example below we will focus on some configuration tasks for TCP/IP and Resolver. Note that this does not cover a complete set up of TCP/IP or Resolver, it just highlights some key aspects that might be applicable to your site.

1. In the JCL below we see that TCP/IP will use SYS1.TCPPARMS(TCPDATA) to determine the stack's hostname.

```
//TCPIP    PROC  PARMS='CTRACE(CTIEZB00)',PROF=TCPPROF,DATA=TCPDATA
//*
//* TCP/IP NETWORK
//*
//TCPIP    EXEC  PGM=EZBTCP,REGION=0M,TIME=1440,PARM=&PARMS
//PROFILE  DD   DISP=SHR,DSN=SYS1.TCPPARMS(&PROF)
//SYSTCPD  DD   DISP=SHR,DSN=SYS1.TCPPARMS(&DATA)
//SYSPRINT DD   SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//ALGPRINT DD   SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CFGPRINT DD   SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSOUT   DD   SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CEEDUMP  DD   SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSERROR DD   SYSOUT=*
```

2. SYS1.TCPPARMS(TCPDATA) tells us that we want the system name to be the hostname and that we don't use a domain name server (DNS); all names will be resolved via site table lookup.

```
; HOSTNAME specifies the TCP host name of this system.  If not
; specified, the default HOSTNAME will be the node name specified
; in the IEFSSNxx PARMLIB member.
;
; HOSTNAME
;
; DOMAINORIGIN specifies the domain origin that will be appended
; to host names passed to the resolver.  If a host name contains
; any dots, then the DOMAINORIGIN will not be appended to the
; host name.
;
DOMAINORIGIN  RALEIGH.IBM.COM
;
; NSINTERADDR specifies the IP address of the name server.
```



```

; LOOPBACK (14.0.0.0) specifies your local name server. If a name
; server will not be used, then do not code an NSINTERADDR statement.
; (Comment out the NSINTERADDR line below). This will cause all names
; to be resolved via site table lookup.
;
; NSINTERADDR 14.0.0.0
;
; TRACE RESOLVER will cause a complete trace of all queries to and
; responses from the name server or site tables to be written to
; the user's console. This command is for debugging purposes only.
;
; TRACE RESOLVER

```

3. In the Resolver JCL we see that the SETUP DD statement is not used. As mentioned in “Understanding resolvers” on page 87, this means that GLOBALTCPIPDATA and other variables will not be used.

```

//RESOLVER PROC PARMS='CTRACE(CTIRES00)'
//*
/* IP NAME RESOLVER – START WITH SUB=MSTR
/*
//RESOLVER EXEC PGM=EZBREINI,REGION=0M,TIME=1440,PARM=&PARMS
/*SETUP DD DISP=SHR,DSN=USER.PROCLIB(RESSETUP),FREE=CLOSE

```

4. If we assume that the RESOLVER_CONFIG environment variable is not set, we can see in Table 13 on page 92 that Resolver will try to use /etc/resolv.conf as base configuration file.

```

TCPIPJOBNAME TCPIP
DomainOrigin RALEIGH.IBM.COM
HostName CDFMVS08

```

As mentioned in “Search orders used in the z/OS UNIX environment” on page 88, the base configuration file contains TCPIP.DATA statements. If the system name is CDFMVS08 (TCPDATA stated that the system name is used as hostname) we can see that /etc/resolv.conf is in sync with SYS1.TCPPARMS(TCPDATA). There are no DNS definitions so site table lookup will be used.

5. Table 13 on page 92 also tells us that if we don’t have to do anything to use the default ASCII-EBCDIC translation table.
6. Assuming that the TSO **MAKESITE** command is not used (can create the X_SITE and X_ADDR variables), /etc/hosts will be the site table used for name lookup.

```

# Resolver /etc/hosts file cdfmvs08
9.42.112.75 cdfmvs08 # CDFMVS08 Host
9.42.112.75 cdfmvs08.raleigh.ibm.com # CDFMVS08 Host
127.0.0.1 localhost

```

The minimal content of this file is information about the current system. In the sample above we define both cdfmvs08 and cdfmvs08.raleigh.ibm.com as a valid name for the IP address of our z/OS system.

If we were using a domain name server (DNS), the DNS would hold the /etc/hosts info, and /etc/resolv.conf and SYS1.TCPPARMS(TCPDATA) would have statements that identify the DNS to our system.

To avoid confusion, it is advised to keep the TCP/IP and Resolver configuration files in sync with each other.

Table 13. Local definitions available to resolver

File type description	APIs affected	Candidate files
Base resolver configuration files	All APIs	<ol style="list-style-type: none"> 1. GLOBALTCPIPDATA 2. RESOLVER_CONFIG environment variable 3. /etc/resolv.conf 4. SYSTCPD DD-name 5. userid.TCPIP.DATA 6. jobname.TCPIP.DATA 7. SYS1.TCPPARMS(TCPDATA) 8. DEFAULTTCPIPDATA 9. TCPIP.TCPIP.DATA
Translate tables	All APIs	<ol style="list-style-type: none"> 1. X_XLATE environment variable 2. userid.STANDARD.TCPXLBIN 3. jobname.STANDARD.TCPXLBIN 4. hlq.STANDARD.TCPXLBIN 5. Resolver-provided translate table, member STANDARD in SEZATCPX
Local host tables	endhostent endnetent getaddrinfo gethostbyaddr gethostbyname gethostent GetHostNumber GetHostResol GetHostString getnameinfo getnetbyaddr getnetbyname getnetent IsLocalHost Resolve sethostent setnetent	IPv4 <ol style="list-style-type: none"> 1. X_SITE environment variable 2. X_ADDR environment variable 3. /etc/hosts 4. userid.HOSTS.xxxxINFO 5. jobname.HOSTS.xxxxINFO 6. hlq.HOSTS.xxxxINFO IPv6 <ol style="list-style-type: none"> 1. GLOBALIPNODES 2. RESOLVER_IPNODES environment variable 3. userid.ETC.IPNODES 4. jobname.ETC.IPNODES 5. hlq.ETC.IPNODES 6. DEFAULTIPNODES 7. /etc/ipnodes

Note: The table above is a partial copy from a table in *Communications Server IP Configuration Guide (SC31-8775)*. See that manual for the full table.

Appendix D. Setting up INETD

This appendix is provided to assist you with some common problems that you may encounter when setting up INETD, or during checking and/or modifying an existing setup.

The INETD daemon provides service management for an IP network. It reduces system load by invoking other daemons only when they are needed and by providing several simple internet services (like echo) internally. INETD reads the `inetd.conf` configuration file to determine which extra services to provide. `ETC.SERVICES` is used to link the services to ports.

`inetd.conf`

The services that rely on INETD are defined in `inetd.conf`, which is read by INETD at startup time. The default location and name of `inetd.conf` is `/etc/inetd.conf`. A sample `inetd.conf` file can be found at `/samples/inetd.conf`.

The following syntax rules apply to `inetd.conf` entries:

- Comments begin with a pound sign (#) or semi-colon (;) and continue until the end of the line
- Entries are case sensitive
- Entries are field-sensitive, but not column sensitive
- Fields are separated with a space or tab character
- Entries can span multiple lines, following these additional syntax rules
 - The split must be in between two separate words (separated by a space or tab character)
 - The continuation line must start with a space or tab character
 - No comments may be imbedded in the continuation

Each entry consists of 7 positional fields, corresponding to the form:

`service_name socket_type protocol wait_flag userid server_program server_program_arguments`

[`ip_address`]:`service_name`

`ip_address` is a local IP, followed by a colon (:). If specified, the address is used instead of `INADDR_ANY` or the current default. To specifically request `INADDR_ANY`, use `"*"`. If `ip_address` (or a colon) is specified without any other entries on the line, it becomes the default for subsequent lines until a new default is specified. `service_name` is a well-known or user-defined service name. The name specified must match one of the server names defined in `ETC.SERVICES`.

`socket_type`

`stream` or `dgram`, to indicate that a stream or datagram socket is used for the service.

`protocol`[,`sndbuf=n`[,`rcvbuf=n`]]

`protocol` can be `tcp[4|6]` or `udp[4|6]`, and is used to further qualify the service name. Both the service name and the protocol must match an entry in `ETC.SERVICES`, except that the "4" or "6" should not be included in the `ETC.SERVICES` entry.

sndbuf and rcvbuf specify the size of the send and receive buffers. The size, represented by n, may be in bytes, or a "k" or "m" may be added to indicate kilobytes or megabytes respectively. sndbuf and rcvbuf can be used in either order.

wait_flag[.max]

wait or nowait.wait indicates the daemon is single-threaded and another request will not be serviced until the first one completes. If nowait is specified, INETD issues an accept when a connect request is received on a stream socket. If wait is specified, it is the responsibility of the server to issue the accept if this is a stream socket.

max is the maximum number of users allowed to request service in a 60 second interval. The default is 40. If exceeded, the service's port is shut down.

userid[.group]

userid is the user ID that the forked daemon is to execute under. This user ID can be different than the INETD user ID. The permissions assigned to this user ID depend on the needs of the service. The INETD user ID needs BPX.DAEMON permission to switch the forked process to this user ID.

The optional group value, which is separated from userid by a dot (.), allows the server to run with a different group ID than the default for this user ID.

server_program

server_program is the full pathname of the service. For example:
/usr/sbin/rlogind is the full pathname for the rlogind command.

server_program_arguments

Maximum of 20 arguments. The first argument is the server name.

ETC.SERVICES

INETD uses ETC.SERVICES to map port numbers and protocols to the services it must support. It can be either a MVS data set or z/OS UNIX file. A sample is shipped in SEZAINST(SERVICES), which is also available as /usr/lpp/tcpip/samples/services. The search order for ETC.SERVICES depends on INETD's startup method; z/OS UNIX or native MVS.

The following syntax rules apply to the services information specification:

- An ETC.SERVICES MVS data set must be fixed or fixed block with an LRECL between 56 and 256
- An ETC.SERVICES HFS file can have a maximum line length of 256
- Items on a line are separated by spaces or tab characters
- Each service is listed on a single line
- A service name must start in the first position on a line
- The maximum service name and alias name length is 32 characters
- A maximum of 35 aliases will be recognized
- Service and alias names are case sensitive
- Comments begin with a pound sign (#) or semi-colon (;) and continue until the end of the line

Each entry consists of 4 positional fields, corresponding to the form:

service_name port_number/protocol aliases

service_name

Specifies a well-known or user-defined service name

port_number

Specifies the socket port number used for the service

protocol

Specifies the transport protocol used for the service. Valid values are tcp and udp

aliases

Specifies a list of unofficial service names

Search order used in the z/OS UNIX environment

The search order used to access ETC.SERVICES in z/OS UNIX is as follows. The search ends at the first file found:

1. **/etc/services**

2. **userid.ETC.SERVICES**

userid is the user ID that is used to start INETD.

3. **hlq.ETC.SERVICES**

hlq represents the value of the DATASETPREFIX statement specified in the base resolver configuration file (if found); otherwise, hlq is TCPIP by default.

Search order used in the native MVS environment

The search order used to access ETC.SERVICES in native MVS is as follows. The search ends at the first data set found:

1. **//SERVICES DD card**

The data set allocated to DD statement SERVICES is used.

2. **userid.ETC.SERVICES**

userid is the user ID that is used to start INETD.

3. **jobname.ETC.SERVICES**

jobname is the name specified on the JOB JCL statement for batch jobs or the procedure name for a started procedure.

4. **hlq.ETC.SERVICES**

hlq represents the value of the DATASETPREFIX statement specified in the base resolver configuration file (if found); otherwise, hlq is TCPIP by default.

Note: Starting INETD through BPXPATCH does not result in using the native MVS search order, since BPXBATCH executes the start command in the z/OS UNIX environment. The native MVS search order is only used when starting a MVS load module, like SEZALOAD(FTP).

PROFILE.TCPIP port definitions

Do not confuse PORT (or PORTRANGE) definitions in PROFILE.TCPIP with ports defined in ETC.SERVICES since these definitions serve different purposes. Ports defined in PROFILE.TCPIP are used by TCPIP to see if the port is reserved for a certain service. ETC.SERVICES is used by INETD to map a port to a service.

When INETD receives a request on a monitored port, it forks a child process (with the requested service) called inetdx, where inetd is the job name for INETD (depends on the startup method) and x is a single digit number.

This complicates port reservation, so if an INETD monitored port is reserved in PROFILE.TCPIP, it is advised to use the name of the started JCL procedure for the z/OS UNIX Kernel Address Space to allow almost any process to bind to the port. This name is typically OMVS, unless a different name is explicitly specified in the STARTUP_PROC parameter of the BPXPRMxx parmlib member.

The following list explains how to determine the job name, given the environment in which the application is run:

- Applications run from batch use the batch job name.
- Applications started from the MVS operator console use the started procedure (STC) name as the job name.
- Applications run from a TSO user ID use the TSO user ID as the job name.
- Applications run from the z/OS shell normally have a job name that is the logged on user ID plus a one-character suffix.
- Authorized users can run applications from the z/OS shell and use the _BPX_JOBNAME environment variable to set the job name. In this case, the value specified for the environment variable is the job name.
- The name of the started JCL procedure for the UNIX System Services Kernel Address Space can be used to allow almost any caller of the bind() socket API (except for users of the Pascal API) to bind to the port. This name is typically OMVS, unless a different name is explicitly specified in the STARTUP_PROC parameter of the BPXPRMxx parmlib member.
- z/OS UNIX applications started by INETD use the job name of the INETD server.

Note: Although it is advised not to do so, ports defined in ETC.SERVICES may differ from the reserved port number for the service in PROFILE.TCPIP.

/etc/inetd.pid

The INETD process creates a temporary file, /etc/inetd.pid, which contains the PID (Process ID) of the currently executing INETD daemon. This PID value is used to identify syslog records that originated from the INETD process, and to provide the PID value for commands that require one, such as kill. It is also used as a lock mechanism to prevent more than 1 INETD process being active.

Startup

The z/OS UNIX implementation of INETD is located by default in /usr/sbin/inetd and supports 2 optional, non-positional, startup parameters:

/usr/sbin/inetd [-d] [inetd.conf]

- d** Debug option. Debug output is written to stderr, which can be routed to a file by the syslogd daemon. Refer to *Communications Server IP Configuration Guide (SC31-8775)* for more information on syslogd. Note that when started this way, INETD will not fork a child process to start a service.

inetd.conf

Configuration file. Default value is /etc/inetd.conf

It is advised to start INETD at IPL time. The most common way to do this is to start it from /etc/rc or /etc/inittab (z/OS 1.8 and higher only). It can also be started from a job or started task using BPXBATCH or from a shell session of a user with appropriate authority.

/etc/rc

When started from the z/OS UNIX initialization shell script, /etc/rc, INETD uses the z/OS UNIX search order to find ETC.SERVICES. A sample /etc/rc file is shipped as /samples/rc. The following sample commands can be used to start INETD:

```
# Start INETD
_BPX_JOBNAME='INETD' /usr/sbin/inetd /etc/inetd.conf &
sleep 5
```

/etc/inittab

z/OS 1.8 and higher provide an alternative method, /etc/inittab, for issuing commands during z/OS UNIX initialization. /etc/inittab allows the definition of the respawn parameter, which restarts the process automatically when it ends (a WTOR is send to the operator for a second restart within 15 minutes). When started from /etc/inittab, INETD uses the z/OS UNIX search order to find ETC.SERVICES. A sample /etc/inittab is shipped as /samples/inittab. The following sample command can be used to start INETD:

```
# Start INETD
inetd::respfrk:/usr/sbin/inetd /etc/inetd.conf
```

Note: Be aware that the respfrk parameter used in the sample will transfer the respawn attribute to all forked processes, including RSE. When the client closes the connection, respawn will start it up again. The RSE server will end again later, due to timeout. Refer to *UNIX System Services Planning (GA22-7800)* to learn more about inittab.

BPXBATCH

The BPXBATCH startup method works both for STC's and user jobs. Note that INETD is a background process, so the BPXBATCH step starting INETD will end within seconds after startup. When started by BPXBATCH, INETD uses the z/OS UNIX search order to find ETC.SERVICES. The JCL listed in Figure 11 is a sample procedure to start INETD (the KILL step removes an active INETD process, if any):

```
//INETD    PROC PRM=
//*
//KILL     EXEC PGM=BPXBATCH,
//          PARM='SH ps -e | grep inetd | cut -c 1-10 | xargs -n 1 kill'
//*
//INETD    EXEC PGM=BPXBATCH,TIME=NOLIMIT,REGION=0M,
//          PARM='PGM /usr/sbin/inetd &PRM'
//STDERR   DD PATH='/tmp/bpxbatch.start.inetd.stderr',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//          PATHMODE=SIRWXU
//* STDIN, STDOUT and STDERR are defaulted to /dev/null
//* INETD daemon output can be controlled by syslogd settings
//*
```

Figure 11. INETD startup JCL

Note: STDIN, STDOUT and STDERR must be z/OS UNIX files when allocated. STDERR can be either a MVS data set or a z/OS UNIX file. Refer to *UNIX System Services Command Reference (SA22-7802)* to learn more about BPXBATCH.

Note: inetd.conf can be a MVS data set or member when INETD is started by BPXBATCH. To do so, code the PARM statement like the following sample (use only single quotes (')):


```
// PARM='PGM /usr/sbin/inetd //'SYS1.TCPPARMS(INETCONF)'' &PRM'
```

Shell session

When started from within a shell session, INETD uses the z/OS UNIX search order to find ETC.SERVICES. The following sample commands can be used (by a person with sufficient authority) to stop and start INETD (# is the z/OS UNIX prompt):

```
1. # ps -e | grep inetd
   7 ?          0:00 /usr/sbin/inetd
2. # kill 7
3. # _BPX_JOBNAME='INETD' /usr/sbin/inetd
```

Note: This method is not advisable for the initial startup, /etc/rc or /etc/inittab are more appropriate since they are executed when z/OS UNIX initializes.

Security

INETD is a z/OS UNIX process and therefore requires valid OMVS definitions in the security software for the user ID associated with INETD. UID, HOME and PROGRAM must be set for the user ID, together with the GID for the user's default group. If INETD is started by /etc/rc or /etc/inittab, the user ID is inherited from the z/OS UNIX kernel, default OMVSKERN.

```
ADDGROUP OMVSGRP OMVS(GID(1))
ADDUSER OMVSKERN DFLTGRP(OMVSGRP) NOPASSWORD +
      OMVS(UID(0) HOME('/') PROGRAM('/bin/sh'))
```

INETD is a daemon that requires access to functions like setuid(). Therefore the user ID used to start INETD requires READ access to the BPX.DAEMON profile in the FACILITY class. If this profile is not defined, UID 0 is mandatory.

```
PERMIT BPX.DAEMON CLASS(FACILITY) ACCESS(READ) ID(OMVSKERN)
```

The INETD user ID also requires EXECUTE permission for the inetd program (/usr/sbin/inetd), READ access to your inetd.conf and ETC.SERVICES file and WRITE access to /etc/inetd.pid. If you want to run INETD without UID 0, you can give CONTROL access to the SUPERUSER.FILESYS profile in the UNIXPRIV class to provide the necessary permits for z/OS UNIX files.

Programs requiring daemon authority must be program controlled if BPX.DAEMON is defined in the FACILITY class. This is already done for the default INETD program (/usr/sbin/inetd), but must be set manually if you use a copy or a custom version. Use the **extattr +p** command to make a z/OS UNIX file program controlled. Use the RACF PROGRAM class to make a MVS load module program controlled.

System programmers who need to restart INETD from within their shell session will start INETD using their permits. Therefore, they must have the same list of permits as the regular INETD user ID. On top of that, they also need permits to list and stop the INETD process. This can be accomplished in multiple ways.

- UID 0
This is not recommended for "human" user IDs since there are no z/OS UNIX related restrictions.
- READ access to the BPX.SUPERUSER profile in the FACILITY class
Allows the user can become UID 0 through the **su** command. This is the recommended setup.
- access to individual profiles that cover the required permissions

- READ access to SUPERUSER.PROCESS.GETPSENT in the UNIXPRIV class (for the **ps** command)
- READ access to SUPERUSER.PROCESS.KILL in the UNIXPRIV class (for the **kill** command)
- READ access to BPX.JOBNAME in the FACILITY class (for the `_BPX_JOBNAME` environment variable)

Refer to *UNIX System Services Command Reference (SA22-7802)* to learn more about the **extattr** and **su** commands. Refer to *UNIX System Services Planning (GA22-7800)* to learn more about the UNIXPRIV class and BPX.* profiles in the FACILITY class. Refer to *Security Server RACF Security Administrator's Guide (SA22-7683)* for more information on the OMVS segment definitions and the PROGRAM class.

Developer for System z requirements

Developer for System z is dependant upon INETD for setting up the client-host connection. It also imposes extra requirements on top of the INETD setup described above.

INETD

INETD's environmental settings, which are passed on when starting a process, and the permissions for INETD's user ID must be set properly in order for INETD to start the RSE server.

- If INETD is started by JCL using BPXBATCH, the region size must be 0.
- If INETD is started from a TSO/OMVS shell session, the TSO region size must be 2096128 or larger.
- If INETD is started by `/etc/rc` or `/etc/inittab`, the region size of SYS1.PROCLIB(BPX0INIT) is used, which is 0 by default.
- READ and EXECUTE access to the Developer for System z installation directories, default `/usr/lpp/wd4z/*`

RSE daemon

The RSE daemon that is started by INETD when a client connects to port 4035 is used to perform authentication, start the RSE server, and return the port number for further communication back to the client. In order to do so, the user ID assigned to the RSE daemon (in `inetd.conf`) requires the following permissions:

- Valid OMVS definitions in the security software; UID, HOME and PROGRAM must be set, together with the GID for the user's default group
- READ access to the BPX.DAEMON profile in the FACILITY class
- READ and EXECUTE access to the Developer for System z installation directories, default `/usr/lpp/wd4z/*`
- READ and EXECUTE access to the Developer for System z configuration directory, default `/usr/lpp/wd4z/rse/lib/`, but it is advised to use a different directory, like `/etc/wd4z/`

Appendix E. Setting up SSL

This appendix is provided to assist you with some common problems that you may encounter when setting up Secure Socket Layer (SSL), or during checking and/or modifying an existing setup.

Secure communication means ensuring that your communication partner is who he claims to be, and transmitting information in a manner that makes it difficult for others to intercept and read the data. Secure Sockets Layer (SSL) provides this ability in a TCP/IP network. It works by using digital certificates to identify yourself and a public key protocol to encrypt the communication. Refer to the *Security Server RACF Security Administrator's Guide (SA22-7683)* for more information on digital certificates and the public key protocol used by SSL.

The actions needed to set up SSL communications for Developer for System z will vary from site to site, depending on the exact needs, the RSE communication method used and what's already available at the site.

In this appendix we will clone the current RSE definitions, so that we have a 2nd connection that will use SSL. Both REXEC and the daemon connection method will be set up for SSL. We will also create our own security certificates to be used by the different parts of the RSE connection. All this will be done in the following steps:

1. Clone the existing RSE setup
2. Determine which key file(s) to use
3. Create a key store with keytool
4. Create a key database (daemon only), with either RACF or gskkyman
5. Activate SSL by updating ssl.properties
6. Test the connection

Note: Refer to the white paper *Setting up SSL for RSE Communication (SC23-5816)* in the Developer for System z internet library, <http://www-306.ibm.com/software/awdtools/devzseries/library/>, for more information on using a certificate signed by a trusted Certificate Authority (CA), or setting up your own CA.

Throughout this appendix, a uniform naming convention is used:

- Certificate : wd4zrse
- Key and certificate storage : wd4zssl.*
- Password : rsessl

Most tasks described below expect you to be active in z/OS UNIX. This can be done by issuing the TSO command **OMVS**. Use the **exit** command to return to TSO.

Clone the existing RSE setup

In this step a new instance of the RSE server and RSE daemon is created to run parallel with the existing one(s). This way, SSL testing will not hinder normal operations. As advised in “Saving the rsed.envvars configuration file in another directory” on page 32, the sample commands below expect the configuration files to be in /etc/wd4z/

```
$ cd /etc/wd4z
$ mkdir ssl
$ cp * ssl
cp: FSUM6254 "ssl" is a directory (not copied)
$ ls ssl
rsecomm.properties  server.zseries      ssl.properties
rsed.envvars        setup.env.zseries
$ cd ssl
$ su
# oedit /etc/services

rsessl          4047/tcp          #Developer for System z RSE using SSL

add rsessl service, using port 4047
# oedit /etc/inetd.conf

rsessl stream tcp nowait OMVSKERN /usr/lpp/wd4z/rse/lib/fekfrsed rsed -d /etc/wd4z/ssl

add rsessl daemon, using configuration directory /etc/wd4z/ssl
# ps -e | grep inetd
7 ?          0:00 /usr/sbin/inetd
# kill 7
# _BPX_JOBNAME='INETD' /usr/sbin/inetd
# exit
$ netstat | grep 4047
INETD4  00016619 0.0.0.0..4047          0.0.0.0..0          Listen
```

The commands listed above create a subdirectory called `ssl` and populate it with the mandatory configuration files. No configuration changes need to be made (yet). We can share the installation directory and MVS components since they are not SSL specific. However, a new daemon (`rsessl`) must be defined that uses the new configuration files. Port 4047 is assigned to the new daemon.

Refer to Chapter 4, “Activating Developer for System z z/OS UNIX components,” on page 31 for more information on the actions shown above.

Determine which key file(s) to use

The identity certificates and the encryption/decryption keys used by SSL are stored in a key file. Different implementations of this key file exist, depending on the application type.

The RSE daemon is a System SSL application, and uses a key database file. This key database can be a physical file created by `gskkyman` or a key ring managed by your security software (e.g. RACF). The RSE server (which is started by the daemon or REXEC) is a Java SSL application, and uses a key store file created by `keytool`. Currently, RACF has no out-of-the-box support for Java SSL.

Thus to connect via REXEC, all we need is the key store file:

- key store (keytool)

To connect via the daemon, we need both the key store and the key database file:

- key store (keytool)
- key database (RACF or gskkyman)

Refer to the *Security Server RACF Security Administrator's Guide (SA22-7683)* for information on RACF and digital certificates. gskkyman documentation can be found in *Cryptographic Services System SSL Programming (SC24-5901)*. keytool documentation is available at <http://java.sun.com/j2se/1.4.2/docs/tooldocs/solaris/keytool.html>.

Create a key store with keytool

"keytool -genkey" Generates a key pair (a public key and associated private key). It then wraps the public key into an X.509 v1 self-signed certificate, which is stored as a single-element certificate chain. This certificate chain and the private key are stored as an entry (identified by an alias) in a (new) key store file.

Note: Java must be included in your command search directories. The following statement might be necessary to be able to execute keytool (/usr/lpp/java/J1.4 is the directory where Java is installed):
PATH=\$PATH:/usr/lpp/java/J1.4/bin

All information can be passed as a parameter, but due to command line length limitations some interactivity is required.

```
$ keytool -genkey -alias wd4zrse -validity 3650 -keystore wd4zssl.jks -storepass
rsessl -keypass rsessl
What is your first and last name?
[Unknown]: wd4z rse ssl
What is the name of your organizational unit?
[Unknown]: wd4z
What is the name of your organization?
[Unknown]: IBM
What is the name of your City or Locality?
[Unknown]: Raleigh
What is the name of your State or Province?
[Unknown]: NC
What is the two-letter country code for this unit?
[Unknown]: US
Is CN=wd4z rse ssl, OU=wd4z, O=IBM, L=Raleigh, ST=NC, C=US correct? (type "yes"
or "no")
[no]: yes
$ ls
rsecomm.properties  server.zseries      ssl.properties
rsed.envvars        setup.env.zseries   wd4zssl.jks
```

The self-signed certificate created above is valid for about 10 years (not counting leap days). It is stored in /etc/wd4z/ssl/wd4zssl.jks using alias wd4zrse. Its password (rsessl) is identical to the key store password, which is a requisite for RSE.

The result can be verified with the -list option:

```
$ keytool -list -alias wd4zrse -keystore wd4zssl.jks -storepass rsessl -v
Alias name: wd4zrse
Creation date: May 24, 2007
Entry type: keyEntry
Certificate chain length: 1
Certificate 1:
Owner: CN=wd4z rse ssl, OU=wd4z, O=IBM, L=Raleigh, ST=NC, C=US
Issuer: CN=wd4z rse ssl, OU=wd4z, O=IBM, L=Raleigh, ST=NC, C=US
Serial number: 46562b2b
```

Valid from: 5/24/07 2:17 PM until: 5/21/17 2:17 PM
Certificate fingerprints:
MD5: 9D:6D:F1:97:1E:AD:5D:B1:F7:14:16:4D:9B:1D:28:80
SHA1: B5:E2:31:F5:B0:E8:9D:01:AD:2D:E6:82:4A:E0:B1:5E:12:CB:10:1C

Create a key database (daemon only)

As mentioned before, the daemon is a System SSL application that uses a key database. This can be either a physical file created by gskkyman or a RACF key ring. RACF key rings are the preferred method for managing private keys and certificates for System SSL.

Note: System SSL uses the Integrated Cryptographic Service Facility (ICSF) if it is available. ICSF provides hardware cryptographic support which will be used instead of the System SSL software algorithms. See *Cryptographic Services System SSL Programming (SC24-5901)* for more information on this.

Create a key ring with RACF

Do not execute this step if you use gskkyman for System SSL.

The **RACDCERT** command installs and maintains private keys and certificates in RACF. RACF supports multiple private keys and certificates to be managed as a group. These groups are called key rings.

Refer to the *Security Server RACF Command Language Reference (SA22-7687)* for details on the **RACDCERT** command.

```
RDEFINE FACILITY IRR.DIGTCERT.LIST UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ACCESS(READ) ID(omvskern)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ACCESS(READ) ID(omvskern)
SETROPTS RACLIST(FACILITY) REFRESH

RACDCERT ID(omvskern) GENCERT SUBJECTSDN(CN('wd4z rse ssl') +
OU('wd4z') O('IBM') L('RaLeigh') SP('NC') C('US')) +
NOTAFTER(DATE(2017-05-21)) WITHLABEL('wd4zrse') KEYUSAGE(HANDSHAKE)

RACDCERT ID(omvskern) ADDRING(wd4zssl.racf)
RACDCERT ID(omvskern) CONNECT(LABEL('wd4zrse') RING(wd4zssl.racf) +
DEFAULT USAGE(PERSONAL))
```

The sample above starts by creating the necessary profiles and permitting user ID OMVSKERN access to key rings. The user ID used must match the user ID coded in /etc/inetd.conf for the SSL RSE daemon. The next step is creating a new, self-signed, certificate with label wd4zrse. No password is needed. This certificate is then added to a newly created key ring (wd4zssl.racf). Just as with the certificate, no password is needed for the key ring. The lifespan of the certificate matches the one created with keytool.

The result can be verified with the list option:

```
RACDCERT ID(omvskern) LIST
Digital certificate information for user OMVSKERN:
```

```
Label: wd4zrse
Certificate ID: 2QjW10Xi0sXZ1aaEqZmihUBA
Status: TRUST
Start Date: 2007/05/24 00:00:00
End Date: 2017/05/21 23:59:59
Serial Number:
```

```

>00<
Issuer's Name:
>CN=wd4z rse ssl.OU=wd4z.O=IBM.L=Raleigh.SP=NC.C=US<
Subject's Name:
>CN=wd4z rse ssl.OU=wd4z.O=IBM.L=Raleigh.SP=NC.C=US<
Private Key Type: Non-ICSF
Private Key Size: 1024
Ring Associations:
Ring Owner: OMVSKERN
Ring:
>wd4zssl.racf<

```

Create a key database with gskkyman

Do not execute this step if you use RACF for System SSL.

gskkyman is a z/OS UNIX shell-based, menu-driven, program that creates, populates and manages a z/OS UNIX file that contains private keys, certificate requests and certificates. This z/OS UNIX file is called a key database.

Note: The following statements might be necessary to set up the environment for gskkyman. See System SSL Programming (SC24-5901) for more information on this.

```

PATH=$PATH:/usr/lpp/gskssl/bin
export NLSPATH=/usr/lpp/gskssl/lib/nls/msg/En_US.IBM-1047/%N:$NLSPATH
export STEPLIB=$STEPLIB:SYS1.SIEALNKE

```

```
$ gskkyman
```

Database Menu

- 1 - Create new database
- 2 - Open database
- 3 - Change database password
- 4 - Change database record length
- 5 - Delete database
- 6 - Create key parameter file

0 - Exit program

Enter option number: 1

Enter key database name (press ENTER to return to menu): wd4zssl.kdb

Enter database password (press ENTER to return to menu): rssl

Re-enter database password: rssl

Enter password expiration in days (press ENTER for no expiration):

Enter database record length (press ENTER to use 2500):

Key database /etc/wd4z/ssl/wd4zssl.kdb created.

Press ENTER to continue.

Key Management Menu

Database: /etc/wd4z/ssl/wd4zssl.kdb

- 1 - Manage keys and certificates
- 2 - Manage certificates
- 3 - Manage certificate requests
- 4 - Create new certificate request
- 5 - Receive requested certificate or a renewal certificate
- 6 - Create a self-signed certificate
- 7 - Import a certificate
- 8 - Import a certificate and a private key
- 9 - Show the default key
- 10 - Store database password

11 - Show database record length

0 - Exit program

Enter option number (press ENTER to return to previous menu): 6

Certificate Type

- 1 - CA certificate with 1024-bit RSA key
- 2 - CA certificate with 2048-bit RSA key
- 3 - CA certificate with 4096-bit RSA key
- 4 - CA certificate with 1024-bit DSA key
- 5 - User or server certificate with 1024-bit RSA key
- 6 - User or server certificate with 2048-bit RSA key
- 7 - User or server certificate with 4096-bit RSA key
- 8 - User or server certificate with 1024-bit DSA key

Select certificate type (press ENTER to return to menu): 5

Enter label (press ENTER to return to menu): wd4zrse

Enter subject name for certificate

Common name (required): wd4z rse ssl

Organizational unit (optional): wd4z

Organization (required): IBM

City/Locality (optional): Raleigh

State/Province (optional): NC

Country/Region (2 characters - required): US

Enter number of days certificate will be valid (default 365): 3650

Enter 1 to specify subject alternate names or 0 to continue: 0

Please wait

Certificate created.

Press ENTER to continue.

Key Management Menu

Database: /etc/wd4z/ssl/wd4zssl.kdb

- 1 - Manage keys and certificates
- 2 - Manage certificates
- 3 - Manage certificate requests
- 4 - Create new certificate request
- 5 - Receive requested certificate or a renewal certificate
- 6 - Create a self-signed certificate
- 7 - Import a certificate
- 8 - Import a certificate and a private key
- 9 - Show the default key
- 10 - Store database password
- 11 - Show database record length

0 - Exit program

Enter option number (press ENTER to return to previous menu): 0

\$ ls -l

total 152

-rwxr-xr-x	1	IBMUSER	SYS1	333	May 24 13:52	rsecomm.properties
-rwxr-xr-x	1	IBMUSER	SYS1	6067	May 24 13:52	rsed.envvars
-rwxr-xr-x	1	IBMUSER	SYS1	332	May 24 13:52	server.zseries
-rwxr-xr-x	1	IBMUSER	SYS1	645	May 24 13:52	setup.env.zseries
-rwxr-xr-x	1	IBMUSER	SYS1	638	May 24 13:52	ssl.properties
-rw-r--r--	1	IBMUSER	SYS1	1224	May 24 14:17	wd4zssl.jks
-rw-----	1	IBMUSER	SYS1	35080	May 24 14:24	wd4zssl.kdb
-rw-----	1	IBMUSER	SYS1	80	May 24 14:24	wd4zssl.rdb

\$ chmod 644 wd4zssl.kdb

\$ chmod 644 wd4zssl.rdb


```
$ ls -l
total 152
-rwxr-xr-x 1 IBMUSER SYS1      333 May 24 13:52 rsecomm.properties
-rwxr-xr-x 1 IBMUSER SYS1    6067 May 24 13:52 rsed.envvars
-rwxr-xr-x 1 IBMUSER SYS1     332 May 24 13:52 server.zseries
-rwxr-xr-x 1 IBMUSER SYS1     645 May 24 13:52 setup.env.zseries
-rwxr-xr-x 1 IBMUSER SYS1     638 May 24 13:52 ssl.properties
-rw-r--r-- 1 IBMUSER SYS1    1224 May 24 14:17 wd4zssl.jks
-rw-r--r-- 1 IBMUSER SYS1   35080 May 24 14:24 wd4zssl.kdb
-rw-r--r-- 1 IBMUSER SYS1      80 May 24 14:24 wd4zssl.rdb
```

The sample above starts by creating a key database called `wd4zssl.kdb` with password `rsessl`. Once the database exists, it is populated by creating a new, self-signed, certificate stored under label `wd4zrse` and with the same password (`rsessl`) as the one used for the key database (this is a RSE requisite).

`gskkyman` allocates the key database with a (very secure) 600 permission bit mask (only owner has access). Unless the daemon uses the same user ID as the creator of the key database, permissions have to be set less restrictive. 640 (owner has read/write, owner's group has read) or 644 (owner has read/write, everyone has read) are usable masks for the `chmod` command.

The result can be verified by selecting the **Show certificate information** option in the **Manage keys and certificates** submenu:

```
$ gskkyman
```

Database Menu

- 1 - Create new database
- 2 - Open database
- 3 - Change database password
- 4 - Change database record length
- 5 - Delete database
- 6 - Create key parameter file
- 0 - Exit program

```
Enter option number: 2
```

```
Enter key database name (press ENTER to return to menu): wd4zssl.kdb
```

```
Enter database password (press ENTER to return to menu): rsessl
```

Key Management Menu

```
Database: /etc/wd4z/ssl/wd4zssl.kdb
```

- 1 - Manage keys and certificates
- 2 - Manage certificates
- 3 - Manage certificate requests
- 4 - Create new certificate request
- 5 - Receive requested certificate or a renewal certificate
- 6 - Create a self-signed certificate
- 7 - Import a certificate
- 8 - Import a certificate and a private key
- 9 - Show the default key
- 10 - Store database password
- 11 - Show database record length
- 0 - Exit program

```
Enter option number (press ENTER to return to previous menu): 1
```

Key and Certificate List

```
Database: /etc/wd4z/ssl/wd4zssl.kdb
```

1 - wd4zrse

0 - Return to selection menu

Enter label number (ENTER to return to selection menu, p for previous list): 1

Key and Certificate Menu

Label: wd4zrse

- 1 - Show certificate information
- 2 - Show key information
- 3 - Set key as default
- 4 - Set certificate trust status
- 5 - Copy certificate and key to another database
- 6 - Export certificate to a file
- 7 - Export certificate and key to a file
- 8 - Delete certificate and key
- 9 - Change label
- 10 - Create a signed certificate and key
- 11 - Create a certificate renewal request
- 0 - Exit program

Enter option number (press ENTER to return to previous menu): 1

Certificate Information

Label: wd4zrse
Record ID: 14
Issuer Record ID: 14
Trusted: Yes
Version: 3
Serial number: 45356379000ac997
Issuer name: wd4z rse ssl
wd4z
IBM
Raleigh
NC
US
Subject name: wd4z rse ssl
wd4z
IBM
Raleigh
NC
US
Effective date: 2007/05/24
Expiration date: 2017/05/21
Public key algorithm: rsaEncryption
Public key size: 1024
Signature algorithm: sha1WithRsaEncryption
Issuer unique ID: None
Subject unique ID: None
Number of extensions: 3

Enter 1 to display extensions, 0 to return to menu: 0

Key and Certificate Menu

Label: wd4zrse

- 1 - Show certificate information
- 2 - Show key information
- 3 - Set key as default
- 4 - Set certificate trust status
- 5 - Copy certificate and key to another database

- 6 - Export certificate to a file
- 7 - Export certificate and key to a file
- 8 - Delete certificate and key
- 9 - Change label
- 10 - Create a signed certificate and key
- 11 - Create a certificate renewal request
- 0 - Exit program

Enter option number (press ENTER to return to previous menu): 0

Activate SSL by updating ssl.properties

Now that the certificates are in place, RSE can start using SSL. Depending on the definitions chosen in the steps above, different values must be provided in `ssl.properties`.

```
$ oedit ssl.properties
```

```
enable_ssl=true
```

Valid values are true and false (default).

```
daemon_keydb_file=wd4zssl.racf
```

gskkyman key database name or RACF key ring name. Only needed for daemon usage.

```
daemon_keydb_password=
```

gskkyman key database password or blank for RACF key ring. Only needed for daemon usage.

```
daemon_key_label=wd4zrse
```

gskkyman/RACF label used, if it is not defined as the default one. Must be commented out if the default is used. Only needed for daemon usage.

```
server_keystore_file=wd4zssl.jks
```

keytool key store name.

```
server_keystore_password=rsessl
```

keytool key store password.

Test the connection

The SSL host setup is now complete and can be tested by connecting with the Developer for System z client. Since we created a new configuration for use by SSL (by cloning the existing one), a new connection must be defined, using following characteristics:

- REXEC: using path `/etc/wd4z/ssl`
- daemon: using port 4047

Note: In order to run a System SSL application (daemon connection), `SYS1.SIEALNKE` must be in `LINKLIST` or `STEPLIB`. If you prefer the `STEPLIB` method, add the following statement to the end of `rsed.envvars`. Be aware however that using `STEPLIB` in z/OS UNIX has a negative performance impact, as described in “Avoid use of `STEPLIB`” on page 63.

- If the last `STEPLIB` directive defined earlier in `rsed.envvars` equals `STEPLIB=NONE`
`STEPLIB=SYS1.SIEALNKE`
- If the last `STEPLIB` directive defined earlier in `rsed.envvars` does not equal `STEPLIB=NONE`

STEPLIB=\$STEPLIB:SYS1.SIEALNKE

Upon connection, the host and client will start with some handshaking in order to set up a secure path. Part of this handshaking is the exchange of certificates. If the Developer for System z client does not recognize the host certificate it will prompt the user asking if this certificate can be trusted.



Figure 12. Import Host Certificate

By clicking the Finish button the user can accept this certificate as trusted, after which the connection initialization continues.

Note: The daemon connection uses 2 certificate locations (System SSL and Java SSL), resulting in 2 different certificates and thus 2 confirmations.

Once a certificate is known to the client, this dialog is not shown again. The list of trusted certificates can be managed by selecting **Window > Preferences... > Remote Systems > SSL**, which shows the following dialog:

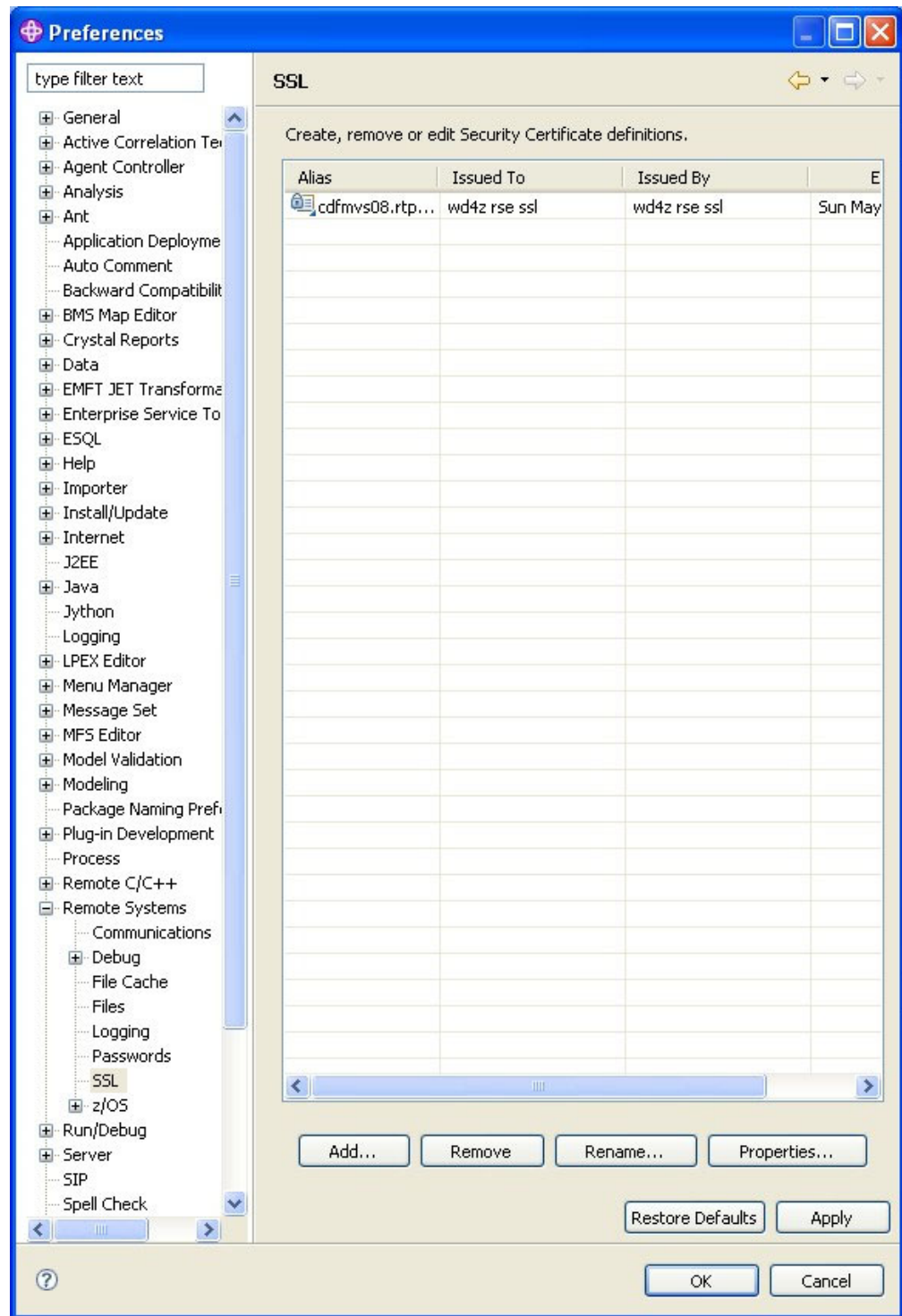


Figure 13. Preferences

If SSL communication fails, the client will return an error message. More information is available in the different log files (`home/.eclipse/RSE/USERID/*` and `/tmp/rsedaemon.log`), as described in “RSE logging” on page 74.

Appendix F. Setting up APPC

This appendix is provided to assist you with some common problems that you may encounter when setting up APPC (Advanced Program-to-Program Communication), or during checking and/or modifying an existing setup.

Refer to *MVS Planning APPC/MVS Management (SA22-7599)* and *MVS Initialization and Tuning Reference (SA22-7592)* for additional information on APPC management and the parmlib members discussed below.

Note that this does not cover a complete set-up of APPC, it just highlights some key aspects that might be applicable to your site.

Member SYS1.SAMPLIB(ATBALL) contains a list and descriptions of all APPC related (sample) members in SYS1.SAMPLIB.

VSAM

APPC/MVS stores its configuration data in SYS1.PARMLIB members and two VSAM data sets:

- The Transaction Program (TP) VSAM data set (default name SYS1.APPCTP) contains scheduling and security information for z/OS programs.
- The Side Information (SI) VSAM data set (default name SYS1.APPCSI) contains the translation of symbolic destination names used by z/OS local TPs and APPC/MVS servers.

A TP is an application program that uses APPC to communicate with a TP on the same or another system to access resources. The APPC setup for Developer for System z activates a new TP called FEKFRSRV, which is referred to as the TSO Commands service.

The sample job listed in figure 14 is a concatenation of sample members SYS1.SAMPLIB(ATBTPVSM) and SYS1.SAMPLIB(ATBSIVSM), and can be used to define the APPC VSAMs.

```

//APPCVSAM JOB <job parameters>
//*
/* CAUTION: This is neither a JCL procedure nor a complete job.
/* Before using this sample, you will have to make the following
/* modifications:
/* 1. Change the job parameters to meet your system requirements.
/* 2. Change ***** to the volume that will hold the APPC VSAMs.
//*
//TP      EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN   DD *
        DEFINE CLUSTER (NAME(SYS1.APPCTP) -
                        VOLUME(*****)) -
                        INDEXED REUSE -
                        SHAREOPTIONS(3 3) -
                        RECORDSIZE(3824 7024) -
                        KEYS(112 0) -
                        RECORDS(300 150)) -
        DATA      (NAME(SYS1.APPCTP.DATA)) -
        INDEX      (NAME(SYS1.APPCTP.INDEX))
//*
//SI      EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN   DD *
        DEFINE CLUSTER (NAME(SYS1.APPCSI) -
                        VOLUME(*****)) -
                        INDEXED REUSE -
                        SHAREOPTIONS(3 3) -
                        RECORDSIZE(248 248) -
                        KEYS(112 0) -
                        RECORDS(50 25)) -
        DATA      (NAME(SYS1.APPCSI.DATA)) -
        INDEX      (NAME(SYS1.APPCSI.INDEX))
//

```

Figure 14. JCL to create APPC VSAMs

VTAM

APPC is an implementation of the Systems Network Architecture (SNA) LU 6.2 protocol. SNA provides formats and protocols that define a variety of physical and logical SNA components, like the Logical Unit (LU). LU 6.2 is a type of logical unit that is specifically designed to handle communications between application programs.

In order to use SNA on MVS, you need to install and configure VTAM (Virtual Telecommunications Access Method). VTAM must be active before the APPC system tasks can be used.

The APPC specific part of the VTAM setup consists of three steps:

1. Define the mode-name used (for example APPCHOST) to VTAM by using SYS1.SAMPLIB(ATBLJOB) to assemble and link-edit SYS1.SAMPLIB(ATBLMODE) into your SYS1.VTAMLIB. See member SYS1.SAMPLIB(ATBLMODE) for details.
2. Define APPC/MVS as a VTAM application by copying sample member SYS1.SAMPLIB(ATBAPPL) to a dataset in the SYS1.VTAMLST concatenation. See member SYS1.SAMPLIB(ATBAPPL) for details.
3. Use console command **v net,act,id=atbappl** to activate the newly defined application (where net equals the name of your VTAM STC). Use console

command **d net,appls** to verify that the application is active. Add the member name to SYS1.VTAMLST(ATCCONxx) if you want it to be activated when VTAM starts.

The ACBNAME of MVSLU01 used in sample member SYS1.SAMPLIB(ATBAPPL) can be changed to match site standards, but must match the definitions in the SYS1.PARMLIB(APPCPMxx) member.

```

MVSLU01 APPL  ACBNAME=MVSLU01,      C
                APPC=YES,            C
                AUTOSES=0,           C
                DDRAINL=NALLOW,      C
                DLOGMOD=APPCHOST,     C
                DMINWNL=5,            C
                DMINWNR=5,            C
                DRESPL=NALLOW,        C
                DSESLIM=10,           C
                LMDENT=19,            C
                MODETAB=LOGMODES,     C
                PARSESS=YES,          C
                SECACPT=CONV,         C
                SRBEXIT=YES,          C
                VPACING=1              C

```

Figure 15. SYS1.SAMPLIB(ATBAPPL)

Refer to the *Communications Server bookshelf (F1A1BK61 for z/OS 1.7)* for more information on configuring VTAM.

SYS1.PARMLIB(APPCPMxx)

To enable and support the flow of conversations between systems, sites must define LUs (Logical Units) between which sessions can bind. A site needs to define at least one LU before APPC/MVS processing can take place, even when APPC processing remains on a single system. LUs are some of the definitions done in SYS1.PARMLIB(APPCPMxx).

The TSO Commands service requires that APPC is set up to have a base LU that can handle both inbound and outbound requests.

The LU definition must be added to the SYS1.PARMLIB(APPCPMxx) member and needs to include the BASE and SCHED(ASCH) parameters. The APPCPMxx member also specifies which transaction profile (TP) and side information (SI) VSAM data sets will be used.

Figure 16 is a sample SYS1.PARMLIB(APPCPMxx) member that can be used for the TSO Commands service.

```

LUADD
  ACBNAME(MVSLU01)
  BASE
  SCHED(ASCH)
  TPDATA(SYS1.APPCTP)
  SIDEINFO DATASET(SYS1.APPCSI)

```

Figure 16. SYS1.PARMLIB(APPCPMxx)

When a system has multiple LU names, you might have to make changes depending on which LU the system selects as the BASE LU. The BASE LU for the system is determined by:

1. The system base LU is represented by the last LUADD statement that contains both the NOSCHED and BASE parameters. This type of system base LU allows outbound requests to be processed when no transaction schedulers are active.
2. If no LUADD statements contain both NOSCHED and BASE, the system base LU is represented by the last LUADD statement that contains the BASE parameter and specifies ASCH as APPC/MVS transaction scheduler. This can be done by either coding SCHED(ASCH) or not coding the SCHED parameter at all (ASCH is the default value for SCHED).

If your system has a LU with BASE and NOSCHED parameters, this LU would be used as the BASE LU but the TSO Command service will not work because this LU does not have a transaction scheduler to handle requests to the FEKFRSRV transaction. If this LU cannot be changed to remove the NOSCHED parameter, the `rsed.envvars` environment variable `_FEKFSCMD_PARTNER_LU` can be set to the LU that has BASE and SCHED(ASCH), such as:

```
_FEKFSCMD_PARTNER_LU=MVSLU01
```

See “Customize `rsed.envvars`, the configuration file for RSE” on page 33 for more information on `rsed.envvars`.

SYS1.PARMLIB(ASCHPMxx)

The APPC/MVS transaction scheduler (default name is ASCH) initiates and schedules transaction programs in response to inbound requests for conversations. Member `SYS1.PARMLIB(ASCHPMxx)` controls its functioning, for example, with transaction class definitions.

The APPC transaction class used for the TSO Commands service must have enough APPC initiators to allow one initiator for each user of Developer for System z.

Note: There is a difference between APPC and z/OS (JES) initiators. When a Developer for System z client connects to the host, the TSO Commands server is started using the APPC initiator. Developer for System z uses a z/OS (JES) initiator when a project is build, a remote syntax check is done or when a job is submitted.

The TSO Commands service also needs the default specifications to be specified in the `OPTIONS` and `TPDEFAULT` sections.

Figure 17 on page 117 is a sample `SYS1.PARMLIB(ASCHPMxx)` member that can be used for the TSO Commands service.

```

CLASSADD
  CLASSNAME(A)
  MAX(20)
  MIN(1)
  MSGLIMIT(200)

OPTIONS
  DEFAULT(A)

TPDEFAULT
  REGION(2M)
  TIME(5)
  MSGLEVEL(1,1)
  OUTCLASS(X)

```

Figure 17. *SYS1.PARMLIB(ASCHPMxx)*

Note: For debugging purposes, the IBM support center might ask you to increase the value of MSGLIMIT, so that more output is written to the log file.

Activating APPC changes

The configuration changes documented in the steps above can now be activated. This can be done in various ways, depending on the circumstances:

- APPC isn't active yet. Enter the following console commands to start APPC/MVS (where xx equals the last 2 characters of the related SYS1.PARMLIB members):

1. S APPC,SUB=MSTR,APPC=xx
2. S ASCH,SUB=MSTR,ASCH=xx

Add these commands to SYS1.PARMLIB(COMMNDxx) to start them at system startup.

- APPC is already active. APPC can dynamically reload the SYS1.PARMLIB members by using a console SET command (where xx equals the last 2 characters of the related SYS1.PARMLIB members):

1. SET APPC=xx
2. SET ASCH=xx

Console commands **D APPC** and **D ASCH** can be used to verify the APPC setup. Refer to *MVS System Commands (SA22-7627)* for more information on the mentioned console commands.

Defining the TSO Commands service transaction

Once APPC/MVS is active, the Developer for System z TSO Commands service can be defined, as described in "(Optional) Define an APPC transaction for the TSO Commands service" on page 23.

The documented way to define the APPC transaction is by customizing and submitting hlq.SFEKSAMP(FEKAPPCC), where hlq equals the high level qualifier used during the installation of Developer for System z (default FEK).

The APPC transaction can also be defined interactively through the APPC ISPF interface, which is documented in a whitepaper. This whitepaper also describes how to set up the APPC transaction to collect user specific accounting information.

The *APPC and WebSphere Developer for System z (SC23-5885)* whitepaper is available at the Developer for System z internet library, <http://www-306.ibm.com/software/awdtools/devzseries/library/>

Note: The Transaction Program (TP) JCL that is used by APPC to start the TSO Commands service has changed in Developer for System z version 7.1. If you follow the directions in the whitepaper to define the TP, you must add the NESTMACS keyword to the PARM line, for example:

```
// PARM='ISPSTART CMD(%FEKFRSRV TIMEOUT=60) NEWAPPL(ISR) NESTMACS'
```

Glossary

A

Action ID. A numeric identifier for an action between 0 and 999

Application Server.

1. A program that handles all application operations between browser-based computers and an organization's back-end business applications or databases. There is a special class of Java-based appservers that conform to the J2EE standard. J2EE code can be easily ported between these appservers. They can support JSPs and servlets for dynamic Web content and EJBs for transactions and database access.
2. The target of a request from a remote application. In the DB2 environment, the application server function is provided by the distributed data facility and is used to access DB2 data from remote applications.
3. A server program in a distributed network that provides the execution environment for an application program.
4. The target of a request from an application requester. The database management system (DBMS) at the application server site provides the requested data.
5. Software that handles communication with the client requesting an asset and queries of the Content Manager.

B

Bidirectional (bi-di). Pertaining to scripts such as Arabic and Hebrew that generally run from right to left, except for numbers, which run from left to right. This definition is from the Localization Industry Standards Association (LISA) Glossary.

Bidirectional Attribute. Text type, text orientation, numeric swapping, and symmetric swapping.

Build Request. A request from the client to perform a build transaction.

Build Transaction. A job started on MVS to perform builds after a build request has been received from the client.

C

Compile.

1. In Integrated Language Environment (ILE) languages, to translate source statements into modules that then can be bound into programs or service programs.
2. To translate all or part of a program expressed in a high-level language into a computer program expressed in an intermediate language, an assembly language, or a machine language.

Container.

1. In CoOperative Development Environment/400, a system object that contains and organizes source files. An i5/OS[®] library or an MVS-partitioned data set are examples of a container.
2. In J2EE, an entity that provides life-cycle management, security, deployment, and run-time services to components. (Sun) Each type of container (EJB, Web, JSP, servlet, applet, and application client) also provides component-specific services
3. In Backup Recovery and Media Services, the physical object used to store and move media such as a box, a case, or a rack.
4. In a virtual tape server (VTS), a receptacle in which one or more exported logical volumes (LVOLs) can be stored. A stacked volume containing one or more LVOLs and residing outside a VTS library is considered to be the container for those volumes.
5. A physical storage location of the data. For example, a file, directory, or device.
6. A column or row that is used to arrange the layout of a portlet or other container on a page.
7. An element of the user interface that holds objects. In the folder manager, an object that can contain other folders or documents.

D

Database. A collection of interrelated or independent data items that are stored together to serve one or more applications.

Data Definition View. Contains a local representation of databases and their objects and provides features to manipulate these objects and export them to a remote database

Data Set. The major unit of data storage and retrieval, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access.

Debug. To detect, diagnose, and eliminate errors in programs.

Debugging Session. The debugging activities that occur between the time that a developer starts a debugger and the time that the developer exits from it.

E

Error Buffer. A portion of storage used to hold error output information temporarily.

F

.

G

Gateway.

1. A middleware component that bridges Internet and intranet environments during Web service invocations.
2. Software that provides services between the endpoints and the rest of the Tivoli® environment.
3. A component of a Voice over Internet Protocol that provides a bridge between VoIP and circuit-switched environments.
4. A device or program used to connect networks or systems with different network architectures. The systems may have different characteristics, such as different communication protocols, different network architecture, or different security policies, in which case the gateway performs a translation role as well as a connection role.

H

.

I

Interactive System Productivity Facility (ISPF). An IBM licensed program that serves as a full-screen editor and dialog manager. Used for writing application programs, it provides a means of generating standard screen panels and interactive dialogs between the application programmer and terminal user. ISPF consists of four major components: DM, PDF, SCLM, and C/S. The DM component is the Dialog Manager, which provides services to dialogs and end-users. The PDF component is the Program Development Facility, which provides services to assist the dialog or application developer. The SCLM component is the Software Configuration Library Manager, which provides services to application developers to manage their application development libraries. The C/S component is the Client/Server, which allows you to run ISPF on programmable workstation, to display the panels using the display function of your workstation

operating system, and to integrate workstation tools and data with host tools and data.

Interpreter. A program that translates and runs each instruction of a high-level programming language before it translates and runs the next instruction.

Isomorphic. Each composed element (in other words, an element containing other elements) of the XML instance document starting from the root has one and only one corresponding COBOL group item whose nesting depth is identical to the nesting depth of its XML equivalent. Each non-composed element (in other words, an element that does not contain other elements) in the XML instance document starting from the top has one and only one corresponding COBOL elementary item whose nesting depth is identical to the nesting level of its XML equivalent and whose memory address at runtime can be uniquely identified.

J

.

K

.

L

Linkage Section. The section in the data division of an activated unit (a called program or an invoked method) that describes data items available from the activating unit (a program or a method). These data items can be referred to by both the activated unit and the activating unit.

Load Library. A library containing load modules.

Lock Action. Locks a member

M

.

N

Navigator View. Provides a hierarchical view of the resources in the Workbench.

Non-Isomorphic. a simple mapping of COBOL items and XML elements belonging to XML documents and COBOL groups that are not identical in shape (non-isomorphic). Non-isomorphic mapping can also be created between non-isomorphic elements of isomorphic structures.

O

Output Console View. Displays the output of a process and allows you to provide keyboard input to a process.

Output View. Displays messages, parameters, and results that are related to the objects that you work with

P

Perspective. A group of views that show various aspects of the resources in the workbench. The workbench user can switch perspectives, depending on the task at hand, and customize the layout of views and editors within the perspective.

Q

.

R

RAM. Repository Access Manager

Remote File System. A file system residing on a separate server or operating system.

Remote System. Any other system in the network with which your system can communicate.

Remote Systems Perspective. Provides an interface for managing remote systems using conventions that are similar to ISPF

Repository.

1. A storage area for data. Every repository has a name and an associated business item type. By default, the name will be the same as the name of the business item. For example, a repository for invoices will be called Invoices. There are two types of information repositories: local (specific to the process) and global (reusable).
2. A VSAM data set on which the states of BTS processes are stored. When a process is not executing under the control of BTS, its state (and the states of its constituent activities) are preserved by being written to a repository data set. The states of all processes of a particular process-type (and of their activity instances) are stored on the same repository data set. Records for multiple process-types can be written to the same repository.
3. A persistent storage area for source code and other application resources. In a team programming environment, a shared repository enables multi-user access to application resources.
4. A collection of information about the queue managers that are members of a cluster. This

information includes queue manager names, their locations, their channels, what queues they host, and so on.

Repository Instance. A project or component that exists in an SCM.

Repositories View. Displays the CVS repository locations that have been added to your Workbench.

Response File.

1. A file that contains a set of predefined answers to questions asked by a program and that is used instead of entering those values one at a time.
2. An ASCII file that can be customized with the setup and configuration data that automates an installation. The setup and configuration data would have to be entered during an interactive install, but with a response file, the installation can proceed without any intervention.

S

Servers View. Displays a list of all your servers and the configurations that are associated with them.

Shell. A software interface between users and the operating system that interprets commands and user interactions and communicates them to the operating system. A computer may have several layers of shells for various levels of user interaction.

Shell Name. The name of the shell interface.

Shell Script. A file containing commands that can be interpreted by the shell. The user types the name of the script file at the shell command prompt to make the shell execute the script commands.

Sidedeck. A library that publishes the functions of a DLL program. The entry names and module names are stored in the library after the source code is compiled.

Silent Installation. An installation that does not send messages to the console but instead stores messages and errors in log files. Also, a silent installation can use response files for data input.

Silent Uninstallation. An uninstallation process that does not send messages to the console but instead stores messages and errors in log files after the uninstall command has been invoked.

T

Task List. A list of procedures that can be executed by a single flow of control.

U

URL. Uniform Resource Locator

V

.

W

.

X

.

Y

.

Z

.

Notices

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*

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 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.

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.

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

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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples 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:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. 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
- CICSplex
- DB2
- IBM
- IMS
- MVS
- OS/390®
- RACF
- Rational®
- System z™
- VTAM
- 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 Rational Developer for System z
Host Configuration Guide
Version 7.1.1

Publication No. SC23-7658-01

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

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. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:

- Send your comments to the address on the reverse side of this form.
- Send a fax to the following number: 1-800-227-5088 (US and Canada)
- Send your comments via e-mail to: kfrye@us.ibm.com

If you would like a response from IBM, please fill in the following information:

Name

Address

Company or Organization

Phone No.

E-mail address



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 G71A / Bldg. 503
P.O. Box 12195
Research Triangle Park, NC
27709-2195



Fold and Tape

Please do not staple

Fold and Tape

Cut or Fold
Along Line



Program Number: 5724-T07

Printed in USA

SC23-7658-01

