

Rational IBM Rational Developer for System z
Version 7.6.1

Host Configuration Guide



Rational IBM Rational Developer for System z
Version 7.6.1

Host Configuration Guide



Note

Before using this document, read the general information under "Documentation notices for IBM Rational Developer for System z" on page 325.

Fifth edition (May 2010)

This edition applies to IBM Rational Developer for System z Version 7.6.1 (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

You can fax your comments to: 1-800-227-5088 (US and Canada)

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 IBM Corporation 2005, 2010.

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

Contents

Figures	ix
--------------------------	-----------

Tables	xi
-------------------------	-----------

About this document	xiii
--------------------------------------	-------------

Who should use this document	xiii
Summary of changes	xiii
Description of document content	xiv
Planning	xiv
Basic customization	xiv
(Optional) Common Access Repository Manager (CARMA).	xiv
(Optional) Application Deployment Manager	xiv
(Optional) SCLM Developer Toolkit	xv
(Optional) Other customization tasks	xv
Installation verification	xv
Operator commands	xvi
Troubleshooting configuration problems	xvi
Security considerations	xvi
Understanding Developer for System z	xvi
WLM considerations	xvi
Tuning considerations	xvi
Performance considerations	xvii
CICSTS considerations	xvii
Customizing the TSO environment	xvii
Running multiple instances	xvii
Migration guide	xvii
Setting up SSL and X.509 authentication	xvii
Setting up TCP/IP	xvii
Setting up INETD	xvii
Setting up APPC	xvii
Requisites	xviii

Part 1. Developer for System z customization 1

Chapter 1. Planning. 3

Migration considerations	3
Planning considerations	3
Product overview	3
Skill requirements	4
Time requirements	4
Preinstallation considerations	4
Setup choice	4
Requisite products	5
Required resources	5
Pre-configuration considerations	8
Workload management	8
Resource usage and system limits	8
Required configuration of requisite products	8
User ID considerations	8
Server considerations	9
Configuration method	10
Predeployment considerations	11

Client checklist	11
----------------------------	----

Chapter 2. Basic customization 13

Requirements and checklist	13
Customization setup	13
PARMLIB changes	14
Set z/OS UNIX limits in BPXPRMxx	14
Add started tasks to COMMNDxx.	15
LPA definitions in LPAISTxx	15
APF authorizations in PROGxx.	16
LINKLIST definitions in PROGxx	16
Requisite LINKLIST and LPA definitions	17
LINKLIST definitions for other products.	18
PROCLIB changes	19
JES Job Monitor	19
RSE daemon	19
Lock daemon.	20
JCL limitations for the PARM variable	21
ELAXF* remote build procedures	22
Security definitions	23
FEJJCNFG, JES Job Monitor configuration file	24
rsed.envvars, RSE configuration file	28
Defining the PORTRANGE available for RSE server	36
Defining extra Java startup parameters with _RSE_JAVAOPTS	37
Defining extra Java startup parameters with _RSE_CMDSERV_OPTS	41
ISPF.conf, ISPF's TSO/ISPF Client Gateway configuration file	42
Optional components	43
Installation verification	43

Chapter 3. (Optional) Common Access Repository Manager (CARMA). 45

Requirements and checklist	45
CARMA components	46
CARMA VSAM migration notes	46
RSE interface to CARMA.	47
CARMA server startup using batch submit	49
Adjust CRASRV.properties	49
Adjust CRASUBMT	49
(Optional) Alternative CARMA server startup using CRASTART	50
Adjust CRASRV.properties	51
Adjust crastart.conf.	51
(Optional) Alternative CARMA server startup using TSO/ISPF Client Gateway	53
Adjust CRASRV.properties	53
Adjust ISPF.conf.	53
(Optional) Activating the sample Repository Access Managers (RAMs)	55
Activating the PDS RAM	55
Activating the SCLM RAM	55
Activating the skeleton RAM	56

(Optional) Activating the CA Endeavor SCM RAM	56
Requirements and checklist	56
Define the CA Endeavor SCM RAM	57
CA Endeavor SCM RAM startup using batch	
submit	57
CA Endeavor SCM RAM startup using	
CRASTART	60
(Optional) Customize CRANDVRA	61
(Optional) Customize the CA Endeavor SCM	
RAM	62
(Optional) Supporting multiple RAMs	62
Example	62
(Optional) IRXJCL versus CRAXJCL	63
Create CRAXJCL	64

Chapter 4. (Optional) Application Deployment Manager 65

Requirements and checklist	65
CRD repository	66
CICS administrative utility	66
RESTful versus Web Service	67
CRD server using the RESTful interface	67
CICS primary connection region	67
CICS non-primary connection regions	67
(Optional) Customize CRD server transaction IDs	68
CRD server using the Web Service interface	68
Pipeline message handler	69
CICS primary connection region	69
CICS non-primary connection regions	70
(Optional) Manifest repository	70

Chapter 5. (Optional) SCLM Developer Toolkit 73

Requirements and checklist	73
Prerequisites	73
ISPF.conf updates for SCLMDT	74
rsed.envvars updates for SCLMDT	75
(Optional) Long/short name translation	75
Create LSTRANS.FILE, the long/short name	
translation VSAM	76
rsed.envvars updates for long/short name	
translation	78
(Optional) Install and customize Ant	78
SCLM updates for SCLMDT	79
Remove old files from WORKAREA	80

Chapter 6. (Optional) Other customization tasks 81

(Optional) DB2 stored procedure	81
Workload Manager (WLM) changes	81
PROCLIB changes	82
DB2 changes	82
(Optional) Enterprise Service Tools (EST) support	83
(Optional) CICS bidirectional language support	84
(Optional) Diagnostic IRZ error messages	84
(Optional) RSE SSL encryption	85
(Optional) RSE tracing	88
(Optional) Host based property groups	89
(Optional) Host based projects	90
(Optional) File Manager integration	91

(Optional) Uneditable characters	92
(Optional) Using REXEC (or SSH)	93
Remote (host-based) actions for z/OS UNIX	
subprojects	94
Alternate RSE connection method	94
REXEC (or SSH) set up	94
(Optional) APPC transaction for the TSO	
Commands service	95
Preparation	95
Implementation	97
APPC usage considerations	98
(Optional) WORKAREA cleanup	98

Chapter 7. Installation verification . . . 99

Verify started tasks	99
JMON, JES Job Monitor	99
LOCKD, Lock daemon	99
RSED, RSE daemon	99
Verify services	101
IVP initialization	102
Port availability	103
TCP/IP setup	103
RSE daemon connection	104
JES Job Monitor connection	105
Lock daemon connection	105
ISPF's TSO/ISPF Client Gateway connection	106
(Optional) TSO Commands service connection	
using APPC	107
(Optional) SCLMDT connection	108
(Optional) REXEC connection	109
(Optional) REXEC/SSH shell script	110

Part 2. Developer for System z information. 113

Chapter 8. Operator commands . . . 115

Start (S)	115
JES Job Monitor	115
RSE daemon	115
Lock daemon	116
Modify (F)	117
JES Job Monitor	117
RSE daemon	118
Lock daemon	121
Stop (P)	121
Console messages	121
JES Job Monitor	121
RSE daemon, RSE thread pool server, and lock	
daemon	122
How to read a syntax diagram	123
Symbols	123
Operands	124
Syntax example	124
Nonalphanumeric characters and blank spaces	124
Selecting more than one operand	124
Longer than one line	124
Syntax fragments	125

Chapter 9. Troubleshooting configuration problems 127

Log and setup analysis using FEKLOGS	127
Log files	128
JES Job Monitor logging	129
Lock daemon logging	129
RSE daemon and thread pool logging	129
RSE user logging	130
Fault Analyzer Integration logging	131
File Manager Integration logging	131
SCLM Developer Toolkit logging	131
CARMA logging	131
APPC transaction (TSO Commands service) logging	132
fekfivpi IVP test logging	132
fekfivps IVP test logging	132
Dump files	133
MVS dumps	133
Java dumps	133
z/OS UNIX dump locations	134
Tracing	135
JES Job Monitor tracing	135
RSE tracing	135
Lock daemon tracing	136
CARMA tracing	136
Error feedback tracing	136
z/OS UNIX permission bits	137
SETUID file system attribute	137
Program Control authorization	137
APF authorization	139
Sticky bit	139
Reserved TCP/IP ports	140
Address Space size	141
startup JCL requirements	141
Limitations set in SYS1.PARMLIB(BPXPRMxx)	141
Limitations stored in the security profile	142
Limitations enforced by system exits	142
Limitations for 64-bit addressing	142
APPC transaction and TSO Commands service	142
Miscellaneous information	144
System limits	144
Known requisite issues	144
Host Connect Emulator	145

Chapter 10. Security considerations 147

Authentication methods	147
User ID and password	148
User ID and one-time password	148
X.509 certificate	148
JES Job Monitor authentication	148
Connection security	148
Limit external communication to specified ports	149
Communication encryption using SSL	149
Port Of Entry checking	149
TCP/IP ports	150
External communication	150
Internal communication	151
CARMA and TCP/IP ports	151
Using PassTickets	152
Audit logging	152

Audit control	153
Audit data	153
JES security	153
Actions against jobs - target limitations	153
Actions against jobs - execution limitations	155
Access to spool files	156
SSL encrypted communication	157
Client authentication using X.509 certificates	158
Certificate Authority (CA) validation	158
(Optional) Query a Certificate Revocation List (CRL)	159
Authentication by your security software	159
Authentication by RSE daemon	160
Port Of Entry (POE) checking	161
CICSTS security	161
CRD repository	161
CICS transactions	162
SSL encrypted communication	162
SCLM security	162
Developer for System z configuration files	162
JES Job Monitor - FEJJCENFG	162
RSE - rsed.envvars	162
RSE - ssl.properties	163
Security definitions	163
Requirements and checklist	164
Activate security settings and classes	165
Define an OMVS segment for Developer for System z users	166
Define data set profiles	166
Define the Developer for System z started tasks	169
Define JES command security	170
Define RSE as a secure z/OS UNIX server	171
Define MVS program controlled libraries for RSE	172
Define application protection for RSE	172
Define PassTicket support for RSE	173
Define z/OS UNIX program controlled files for RSE	174
Verify security settings	174

Chapter 11. Understanding Developer for System z 177

Component overview	177
RSE as a Java application	179
Task owners	180
Connection flow	182
Lock daemon	183
Freeing a lock	184
z/OS UNIX directory structure	185
Update privileges for non-system administrators	186

Chapter 12. WLM considerations . . . 187

Workload classification	187
Classification rules	188
Setting goals	189
Considerations for goal selection	190
STC	191
OMVS	191
JES	192
ASCH	193

I	CICS	194
---	----------------	-----

Chapter 13. Tuning considerations 195

Resource usage	195
Overview	196
Address space count	197
Process count	200
Thread count	202
Storage usage	205
Java heap size limit	205
Address space size limit	205
Size estimate guidelines	206
Sample storage usage analysis	206
z/OS UNIX file system space usage	210
Key resource definitions	212
/etc/rdz/rsed.envvars	212
SYS1.PARMLIB(BPXPRMxx)	213
Various resource definitions	216
EXEC card in the server JCL	216
FEK.#CUST.PARMLIB(FEJJCENFG)	216
SYS1.PARMLIB(IEASYSxx)	216
SYS1.PARMLIB(IVTPRMxx)	216
SYS1.PARMLIB(ASCHPMxx)	217
Monitoring	217
Monitoring RSE	217
Monitoring z/OS UNIX	218
Monitoring the network	220
Monitoring z/OS UNIX file systems	220
Sample setup	220
Thread pool count	221
Determine minimum limits	221
Defining limits	221
Monitor resource usage	222

Chapter 14. Performance considerations 225

Use zFS file systems	225
Avoid use of STEPLIB	225
Improve access to system libraries	225
Language Environment (LE) runtime libraries	225
Application development	226
Improving performance of security checking	226
Workload management	227
Fixed Java heap size	227
Java -Xquickstart option	227
Class sharing between JVMs	228
Enable class sharing	228
Cache size limits	228
Cache security	228
SYS1.PARMLIB(BPXPRMxx)	229
Disk space	229
Cache management utilities	229

Chapter 15. CICSTS considerations 231

RESTful versus Web Service	232
Primary versus non-primary connection regions	232
CICS resource install logging	232
Application Deployment Manager security	233
CRD repository security	233
Pipeline security	233

Transaction security	233
SSL encrypted communication	234
Resource security	234
Administrative utility	235
Administrative utility migration notes	239
Administrative utility messages	240

Chapter 16. Customizing the TSO environment 243

The TSO Commands service	243
Access methods	243
Using the TSO/ISPF Client Gateway access method	243
Basic customization – ISPF.conf	243
Advanced – Use existing ISPF profiles	244
Advanced – Using an allocation exec	244
Advanced – Use multiple allocation execs	245
Advanced – Multiple ISPF.conf files with multiple Developer for System z setups	245
Using the APPC access method	246
Basic customization – APPC transaction JCL	246
Advanced – Use existing ISPF profiles	246
Advanced – Using an allocation exec	247
Advanced – Multiple APPC transactions with multiple Developer for System z setups	247

Chapter 17. Running multiple instances 249

Identical setup across a sysplex	249
Identical software level, different configuration files	250
All other situations	250

Chapter 18. Migration guide 255

Migration considerations	255
Backing up previously configured files	255
Version 7.6.1 migration notes	257
Migrate from version 7.5 to version 7.6	258
IBM Rational Developer for System z, FMID HHOP760	258
Configurable files	260
Migrate from version 7.1 to version 7.5	263
IBM Rational Developer for System z, FMID HHOP750	263
Configurable files	264
Migrate from version 7.0 to version 7.1	266
IBM Rational Developer for System z, FMID HHOP710	266
IBM Common Access Repository Manager (CARMA), FMID HCMA710	266
Configurable files	267

Appendix A. Setting up SSL and X.509 authentication 269

Decide where to store private keys and certificates	269
Create a key ring with RACF	271
(Optional) Using a signed certificate	271
Clone the existing RSE setup	272
Update rsed.envvars to enable coexistence	273
Update ssl.properties to enable SSL	273

Activate SSL by creating a new RSE daemon	273
Test the connection	274
(Optional) Add X.509 client authentication support	277
(Optional) Create a key database with gskkyman	277
(Optional) Create a key store with keytool.	280

Appendix B. Setting up TCP/IP 283

Hostname dependency	283
Understanding resolvers.	284
Understanding search orders of configuration information	284
Search orders used in the z/OS UNIX environment	285
Base resolver configuration files	285
Translate tables.	285
Local host tables	286
Applying this set up information to Developer for System z	286
Host address is not resolved correctly	289

Appendix C. Setting up INETD 291

inetd.conf	291
ETC.SERVICES.	292
Search order used in the z/OS UNIX environment.	293
Search order used in the native MVS environment.	293
PROFILE.TCPIP port definitions	294
/etc/inetd.pid	294
Startup	294
/etc/rc	295
/etc/inittab	295
BPXBATCH	295
Shell session.	296
Security	296
Developer for System z requirements	297
INETD	297
REXEC (or SSH)	298

Appendix D. Setting up APPC 299

VSAM.	299
VTAM.	300
SYS1.PARMLIB(APPCPMxx)	301

SYS1.PARMLIB(ASCHPMxx)	302
Activating APPC changes	303
Defining the TSO Commands service transaction	303
(Optional) Alternative setup options.	303
Alternative transaction name	304
Multiple LUs	304
LU security	304

Appendix E. Requisites 305

z/OS host prerequisites	305
z/OS	305
SMP/E	306
SDK for z/OS Java 2 Technology Edition	307
z/OS host corequisites	307
z/OS	308
COBOL compiler	310
PL/I compiler	310
Debug Tool for z/OS.	311
CICS Transaction Server.	311
IMS	312
DB2 for z/OS	312
Rational Team Concert for System z.	313
File Manager	313
Fault Analyzer	314
REXX	314
Ported tools	314
Ant.	314
Endevor	315

Bibliography. 317

Referenced publications	317
Informational publications	319

Glossary 321

Documentation notices for IBM

Rational Developer for System z 325

Copyright license	326
Trademark acknowledgments	326

Index 329

Figures

1.	JMON - JES Job Monitor started task	19
2.	RSED - RSE daemon started task	20
3.	LOCKD - Lock daemon started task	21
4.	RSED - alternate RSE daemon startup.	21
5.	rsed.stdin.sh - alternate RSE daemon startup	22
6.	FEJJCNGF, JES Job Monitor configuration file	25
7.	rsed.envvars - RSE configuration file	30
8.	(continued).	31
9.	ISPF.conf - ISPF configuration file	42
10.	CRASRV.properties - CARMA configuration file	47
11.	CRASRV.properties - CARMA startup using batch submit	49
12.	CRASUBMT - CARMA startup using batch submit	50
13.	CRASRV.properties - *CRASTART alternative CARMA startup	51
14.	crastart.conf - *CRASTART alternative CARMA startup	52
15.	CRASRV.properties - *ISPF alternative CARMA startup	53
16.	ISPF.conf - *ISPF alternative CARMA startup	54
17.	Figure x1. CRASRV.properties - CA Endeavor SCM RAM startup using batch submit	58
18.	Figure x2. CRASUBMT - CA Endeavor SCM RAM startup using batch submit	59
19.	Figure x3. CRASRV.properties - CA Endeavor SCM RAM startup using CRASTART	60
20.	crastart.conf - CA Endeavor SCM RAM startup using CRASTART	61
21.	ISPF.conf updates for SCLMDT	75
22.	rsed.envvars updates for SCLMDT.	75
23.	FLM02LST - long/short name translation setup JCL	77
24.	ELAXMSAM - DB2 stored procedure task	82
25.	ELAXMJCL - DB2 stored procedure definition	83
26.	ssl.properties - SSL configuration file	86
27.	rsecomm.properties - Logging configuration file	88
28.	propertiescfg.properties - Host-based property groups configuration file	90
29.	projectcfg.properties - Host-based projects configuration file.	91
30.	FMIEXT.properties - File Manager configuration file.	92
31.	uchars.settings - Uneditable characters configuration file.	93
32.	REXX for APPC ISPF panels	96
33.	START JMON operator command.	115
34.	START RSED operator command	115
35.	START LOCKD operator command	116
36.	MODIFY JMON operator command	117
37.	MODIFY RSED operator command	118
38.	MODIFY LOCKD operator command	121
39.	STOP operator command	121
40.	TCP/IP ports	150
41.	Component overview	177
42.	RSE as a Java application	179
43.	Task owners	180
44.	Connection flow	182
45.	Lock daemon flow	183
46.	z/OS UNIX directory structure	185
47.	WLM classification.	187
48.	Maximum number of address spaces	198
49.	Number of address spaces per client	199
50.	Maximum number of processes	201
51.	Number of processes per client	202
52.	Maximum number of RSE thread pool threads.	204
53.	Maximum number of JES Job Monitor threads	204
54.	Resource usage with 5 logons	207
55.	Resource usage with 5 logons (continued)	208
56.	Resource usage while editing a PDS member	209
57.	z/OS UNIX file system space usage	211
58.	Resource usage of sample setup	223
59.	ADNJSPAU - CICSTS administrative utility	237
60.	FEKAPPCC - create a second APPC transaction	248
61.	RSESSL - RSE daemon user job for SSL	274
62.	Import Host Certificate dialog	275
63.	Preferences dialog - SSL	276
64.	INETD startup JCL	296
65.	JCL to create APPC VSAMs.	300
66.	SYS1.SAMPLIB(ATBAPPL)	301
67.	SYS1.PARMLIB(APPCPMxx)	301
68.	SYS1.PARMLIB(ASCHPMxx)	302

Tables

1. Required resources	5	27. JES2 Job Monitor operator commands	170
2. Optional resources	6	28. JES3 Job Monitor operator commands	171
3. Administrators needed for required tasks	6	29. WLM entry-point subsystems	188
4. Administrators needed for optional tasks	7	30. WLM work qualifiers	188
5. Client checklist - mandatory parts	12	31. WLM workloads	189
6. Client checklist - optional parts	12	32. WLM workloads - STC	191
7. Sample ELAXF* procedures	22	33. WLM workloads - OMVS	191
8. ELAXF* high level qualifier checklist	23	34. WLM workloads - JES	193
9. LIMIT_COMMANDS command permission matrix	27	35. WLM workloads - ASCH	193
10. crastart.conf variables	52	36. WLM workloads - CICS	194
11. Default CRD server transaction IDs	68	37. Common resource usage	196
12. Default CRD server transaction IDs	69	38. User-specific requisite resource usage	196
13. SCLM administrator checklist	79	39. User-specific resource usage	196
14. SSL certificate storage mechanisms	86	40. Address space count	197
15. Valid keystore types.	87	41. Address space limits	199
16. APPC transaction checklist	96	42. Process count	200
17. IVPs for services	101	43. Process limits	202
18. Thread pool error status	119	44. Thread count	202
19. RSE console messages.	122	45. Thread limits	205
20. JAVA_DUMP_TDUMP_PATTERN variables	133	46. Log output directives	212
21. JES Job Monitor console commands	154	47. Version 7.6 customizations	260
22. LIMIT_COMMANDS command permission matrix	154	48. Version 7.5 customizations	264
23. Extended JESSPOOL profiles	154	49. Version 7.1 customizations	267
24. LIMIT_VIEW browse permission matrix	156	50. SSL certificate storage mechanisms	270
25. SSL certificate storage mechanisms	157	51. Local definitions available to resolver	288
26. Security setup variables	164	52. Referenced publications	317
		53. Referenced Web sites	319
		54. Informational publications	319

About this document

This document 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 Version 7.6.1 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*.
- *Common Access Repository Manager* is abbreviated to *CARMA*.
- *Software Configuration and Library Manager Developer Toolkit* is called *SCLM Developer Toolkit*, abbreviated to *SCLMDT*.
- *z/OS UNIX[®] System Services* is called *z/OS UNIX*.
- *Customer Information Control System Transaction Server* is called *CICSTS*, abbreviated to *CICS[®]*.

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

Who should use this document

This document is intended for system programmers installing and configuring IBM Rational Developer for System z Version 7.6.1, FMID HHOP760, on their z/OS host system.

It lists in detail the different steps needed to do a full setup of the product, including some non-default scenarios. To use this document, you need to be familiar with the z/OS UNIX System Services and MVS[™] host systems.

Summary of changes

This section summarizes the changes for *Rational[®] Developer for System z[®] Version 7.6.1 Host Configuration Guide*, SC23-7658-04 (updated May 2010).

Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

This document contains information previously presented in *Rational Developer for System z Version 7.6 Host Configuration Guide*, SC23-7658-03.

New information:

- Corrections and additional information presented in *Rational Developer for System z v7.6 Host Configuration Release Notes[®]* are incorporated.
- Version 7.6.1 specific migration notes. See “Version 7.6.1 migration notes” on page 257.
- Added document overview. See “Description of document content” on page xiv.
- Support for 64-bit Java[™]. See Appendix E, “Requisites,” on page 305.
- Product configuration through ISPF panels. See “Pre-configuration considerations” on page 8.

- New directives in `rsed.envvars`. See “`rsed.envvars`, RSE configuration file” on page 28.
- New proclib members. See “ELAXF* remote build procedures” on page 22.
- New CARMA VSAM layout. See “CARMA VSAM migration notes” on page 46.
- Supporting multiple CARMA RAMs. See “(Optional) Supporting multiple RAMs” on page 62.
- New Application Deployment Manager options. See “Administrative utility” on page 235.
- New Application Deployment Manager VSAM layout. See “Administrative utility migration notes” on page 239
- New operator commands. See Chapter 8, “Operator commands,” on page 115.
- New console messages. See “Console messages” on page 121
- Workload Management information. See Chapter 12, “WLM considerations,” on page 187.

Description of document content

This section summarizes the information presented in this document.

Planning

Use the information in this chapter to plan the installation and deployment of Developer for System z.

Basic customization

The customization steps below are for a basic Developer for System z setup:

- Customization setup
- PARMLIB changes
- PROCLIB changes
- Security definitions
- FEJCNFG, JES Job Monitor configuration file
- `rsed.envvars`, RSE configuration file
- `ISPF.conf`, ISPF’s TSO/ISPF Client Gateway configuration file

(Optional) Common Access Repository Manager (CARMA)

Common Access Repository Manager (CARMA) is a productivity aid for developers who are creating Repository Access Managers (RAMs). A RAM is an Application Programming Interface (API) for z/OS based Software Configuration Managers (SCMs).

In turn, user-written applications can start a CARMA server which loads the RAMS(s) and provides a standard interface to access the SCM.

The IBM® Rational® Developer for System z Interface for CA Endevor® Software Configuration Manager gives Developer for System z clients direct access to CA Endevor® SCM.

(Optional) Application Deployment Manager

Developer for System z uses certain functions of Application Deployment Manager as a common deployment approach for various components. Optional

customization enables more features of Application Deployment Manager and can add the following services to Developer for System z:

- IBM CICS Explorer provides an Eclipse-based infrastructure to view and manage CICS resources and enables greater integration between CICS tools.
- CICS Resource Definition (CRD) client and server provide the following functions:
 - CICS Resource Definition editor
 - Allow application developers to define CICS resources in a limited, controlled, and secure fashion.
 - 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.
 - Miscellaneous CICS development aids
 - Miscellaneous CICS Web Service development aids

(Optional) SCLM Developer Toolkit

SCLM Developer Toolkit 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 has an Eclipse-based plugin 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.

(Optional) Other customization tasks

This section combines a variety of optional customization tasks. Follow the instructions in the appropriate section to configure the desired service.

- DB2 stored procedure
- Enterprise Service Tools (EST) support
- CICS bidirectional language support
- Diagnostic IRZ error messages
- RSE SSL encryption
- RSE tracing
- Host based property groups
- Host based projects
- File Manager integration
- Uneditable characters
- Using REXEC (or SSH)
- APPC transaction for the TSO Commands service
- WORKAREA cleanup

Installation verification

After completing the product customization, you can use the Installation Verification Programs (IVPs) described in this chapter to verify the successful setup of key product components.

Operator commands

This chapter provides an overview of the available operator (or console) commands for Developer for System z.

Troubleshooting configuration problems

This chapter is provided to assist you with some common problems that you may encounter during your configuration of Developer for System z, and has the following sections:

- Log and setup analysis using FEKLOGS
- Log files
- Dump files
- Tracing
- z/OS UNIX permission bits
- Reserved TCP/IP ports
- Address Space size
- APPC transaction and TSO Commands service
- Miscellaneous information

Security considerations

Developer for System z provides mainframe access to users on a non-mainframe workstation. Validating connection requests, providing secure communication between the host and the workstation, and authorizing and auditing activity are therefore important aspects of the product configuration.

Understanding Developer for System z

The Developer for System z host consists of several components that interact to give the client access to the host services and data. Understanding the design of these components can help you make the correct configuration decisions.

WLM considerations

Unlike traditional z/OS applications, Developer for System z is not a monolithic application that can be identified easily to Workload Manager (WLM). Developer for System z consists of several components that interact to give the client access to the host services and data. Some of these services are active in different address spaces, resulting in different WLM classifications.

Tuning considerations

RSE (Remote Systems Explorer) is the core of Developer for System z. To manage the connections and workloads from the clients, RSE is composed of a daemon address space, which controls thread pooling address spaces. The daemon acts as a focal point for connection and management purposes, while the thread pools process the client workloads.

This makes RSE a prime target for tuning the Developer for System z setup. However, maintaining hundreds of users, each using 16 or more threads, a certain amount of storage, and possibly one or more address spaces requires proper configuration of both Developer for System z and z/OS.

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.

CICSTS considerations

This chapter contains information useful for a CICS Transaction Server administrator.

Customizing the TSO environment

This chapter assists you with mimicking a TSO logon procedure by adding DD statements and data sets to the TSO environment in Developer for System z.

Running multiple instances

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 such as TCP/IP ports cannot be shared, so the defaults are not always applicable. Use the information in this chapter to plan the coexistence of the different instances of Developer for System z, after which you can use this configuration guide to customize them.

Migration guide

This section highlights installation and configuration changes compared to previous releases of the product. It also gives some general guidelines to migrate to this release. Refer to the related sections in this manual for more information.

Setting up SSL and X.509 authentication

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 or modifying an existing setup. This appendix also provides a sample setup to support users authenticating themselves with an X.509 certificate.

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 or modifying an existing setup.

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 or modifying an existing setup. INETD is used by Developer for System z for REXEC/SSH functionality.

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 or modifying an existing setup.

Requisites

This appendix lists the host prerequisites and corequisites for this version of Developer for System z.

Part 1. Developer for System z customization

Chapter 1. Planning

Use the information in this chapter, together with the information in Appendix E, “Requisites,” on page 305, to plan the installation and deployment of Developer for System z. The following subjects are described:

- “Migration considerations”
- “Planning considerations”
- “Preinstallation considerations” on page 4
- “Pre-configuration considerations” on page 8
- “Predeployment considerations” on page 11
- “Client checklist” on page 11

Migration considerations

Chapter 18, “Migration guide,” on page 255 describes installation and configuration changes compared to previous releases of the product. Use this information to plan your migration to the current release of Developer for System z.

Notes:

1. 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 IBM WebSphere Studio Enterprise Developer, it is recommended that you save the related customized files BEFORE installing IBM Rational Developer for System z Version 7.6.1. Refer to Chapter 18, “Migration guide,” on page 255 for an overview of files that required customization.
2. Refer to Chapter 17, “Running multiple instances,” on page 249, if you plan on running multiple instances of Developer for System z.

Planning considerations

Product overview

Developer for System z consists of a client, installed on the user’s personal computer, and a server, installed on one or more hosts. This documentation will focus on the host being a z/OS system. However, other operating systems, such as AIX® and Linux® on System z, are also supported.

The client provides developers an Eclipse-based development environment that facilitates a uniform graphical interface to the host, and that, among other things, can offload work from the host to the client, saving resources on the host.

The host portion consists of several permanently active tasks and tasks that are started ad-hoc. These tasks allow the client to work with the various components of your z/OS host, such as MVS data sets, TSO commands, z/OS UNIX files and commands, job submit, and job output.

Developer for System z can also interact with subsystems and other application software on the host, such as CICS, Debug Tool, and Software Configuration Managers (SCMs), if Developer for system z is configured to do so, and if these (corequisite) products are available.

Refer to Chapter 11, “Understanding Developer for System z,” on page 177 to get a basic understanding of the Developer for System z design.

Refer to the Developer for System z website, <http://www-01.ibm.com/software/awdtools/rdz/>, or your local IBM representative to learn more about the functionality offered by Developer for System z.

Skill requirements

Minimal SMP/E skills are needed for a Developer for System z host installation.

The configuration of Developer for System z requires more than the typical system programming permissions and expertise, so assistance from others may be needed. Table 3 on page 6 and Table 4 on page 7 list the administrators needed for the required and optional customization tasks.

Time requirements

The amount of time required to install and configure the Developer for System z host components depends on various factors, such as:

- the current z/OS UNIX and TCP/IP configuration
- the availability of prerequisite software and maintenance
- whether or not OMVS segments are defined for Developer for System z users
- the availability of a user, who has successfully installed the client, to test the install and report any problems that might occur

Experience has shown that the installation and configuration process of the Developer for System z host requires from one to four days to complete. This time requirement is for a clean installation performed by an experienced system programmer. If problems are encountered, or if the required skills are not available, then the setup will take longer.

Preinstallation considerations

Refer to *Program Directory for IBM Rational Developer for System z* (GI11-8298) for detailed instructions on the SMP/E installation of the product.

Note: The file system (HFS or zFS) in which Developer for System z is installed must be mounted with the SETUID permission bit on (this is the system default). Mounting the file system with the NOSETUID parameter will prevent Developer for System z from creating the user’s security environment, and will fail the connection request of the client.

Refer to Chapter 17, “Running multiple instances,” on page 249 if you plan on running multiple instances of Developer for System z.

Setup choice

Developer for System z provides a choice on how to access the TSO Commands service. The choice made here impacts the required configuration of prerequisites. One of the following methods must be selected and configured:

- ISPF’s TSO/ISPF Client Gateway service, which requires a minimum ISPF service level. This is the default method used in the provided samples.
- An APPC transaction (as in pre-version 7.1 releases)

Note: ISPF's TSO/ISPF Client Gateway is also used by SCLM Developer Toolkit and optionally by an alternative startup method for Common Access Repository Manager (CARMA).

Requisite products

Appendix E, "Requisites," on page 305 has a list of prerequisite software that must be installed and operational before Developer for System z 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 Prerequisites* (SC23-7659) in the Developer for System z online library at <http://www-01.ibm.com/software/awdtools/rdz/library/> for an up-to-date list of prerequisite and corequisite products for your version of Developer for System z. Plan ahead to have these requisite products available, as it might take some time, depending on the policies at your site. The key requisites for a basic setup are the following:

- z/OS 1.8 or higher
- ISPF APAR OA29489 (TSO/ISPF Client Gateway)
- Java 5.0 or higher

Note: The PTF for Developer for System z APAR PM07305 must be applied when using a 64-bit version of Java. The PTF is available via the Developer for System z recommend service page, <http://www-01.ibm.com/support/docview.wss?rs=2294&context=SS2QJ2&uid=swg27006335>.

Required resources

Developer for System z requires the allocation of the systems resources listed in Table 1. The resources listed in Table 2 on page 6 are required for optional services. Plan ahead to have these resources available, as it might take some time to get them, depending on the policies at your site.

Note: Developer for System z consists of multiple tasks that communicate with each other and the client. These tasks use various timers to detect communication loss with their partner or partners. This implies that timeout issues can arise (due to lack of CPU time during the timeout window) on systems with a heavy CPU load or incorrect Workload Management (WLM) settings for Developer for system z.

Table 1. Required resources

Resource	Default value	Information
APF authorized data set	FEK.SFEKAUTH	"APF authorizations in PROGxx" on page 16
started task	JMON, RSED, and LOCKD	"Server considerations" on page 9
port for host-confined use	6715	"FEJJCNFG, JES Job Monitor configuration file" on page 24
port for host-confined use	4036	"rsed.envvars, RSE configuration file" on page 28
port for client-host communication	4035	"PROCLIB changes" on page 19

Table 1. Required resources (continued)

Resource	Default value	Information
port range for client-host communication	any available port is used	"Defining the PORTRANGE available for RSE server" on page 36
Application security definition	Universal access READ for FEKAPPL	"Define application protection for RSE" on page 172
PassTicket security definitions	no default	"Define PassTicket support for RSE" on page 173

Table 2. Optional resources

Resource	Default value	Information
LINKLIST data set	FEK.SFEKAUTH and FEK.SFEKLOAD	Chapter 5, "(Optional) SCLM Developer Toolkit," on page 73
LPA data set	FEK.SFEKLPA	Chapter 3, "(Optional) Common Access Repository Manager (CARMA)," on page 45
port range for host-confined use	5227-5326 (100 ports)	Chapter 3, "(Optional) Common Access Repository Manager (CARMA)," on page 45
ports for host-confined use	any available port is used	"(Optional) APPC transaction for the TSO Commands service" on page 95
port for client-host communication	no default	Chapter 4, "(Optional) Application Deployment Manager," on page 65
CICS CSD update	multiple values	Chapter 4, "(Optional) Application Deployment Manager," on page 65
CICS JCL update	FEK.SFEKLOAD	<ul style="list-style-type: none"> Chapter 4, "(Optional) Application Deployment Manager," on page 65 "(Optional) CICS bidirectional language support" on page 84

The configuration of Developer for System z requires more than the typical system programming permissions and expertise, so minimal assistance from others may be needed. Table 3 and Table 4 on page 7 list the administrators needed for the required and optional customization tasks.

Table 3. Administrators needed for required tasks

Administrator	Task	Information
System	Typical system programmer actions are required for all customization tasks	N/A

Table 3. Administrators needed for required tasks (continued)

Administrator	Task	Information
Security	<ul style="list-style-type: none"> • Define OMVS segment for Developer for System z users • Define data set profiles • Define started tasks • Define operator command security • Define z/OS UNIX server profiles • Define application security • Define PassTicket support • Define program controlled data sets • Define program controlled z/OS UNIX files 	Chapter 10, “Security considerations,” on page 147
TCP/IP	Define new TCP/IP ports	“TCP/IP ports” on page 150
WLM	Assign started task goals to the servers and their child processes	Chapter 12, “WLM considerations,” on page 187

Table 4. Administrators needed for optional tasks

Administrator	Task	Information
System	Typical system programmer actions are required for all customization tasks	N/A
Security	<ul style="list-style-type: none"> • Define data set profiles • Define program controlled data sets • Define permission to submit xxx* jobs • Define CICS transaction security • Add certificate for SSL • Define X.509 client certificate support 	<ul style="list-style-type: none"> • Chapter 10, “Security considerations,” on page 147 • Chapter 15, “CICSTS considerations,” on page 231 • Appendix A, “Setting up SSL and X.509 authentication,” on page 269
TCP/IP	Define new TCP/IP ports	“TCP/IP ports” on page 150
SCLM	<ul style="list-style-type: none"> • Define SCLM language translators for JAVA/J2EE support • Define SCLM types for JAVA/J2EE support 	Chapter 5, “(Optional) SCLM Developer Toolkit,” on page 73
CICS TS	<ul style="list-style-type: none"> • Update CICS region JCL • Update CICS region CSD • Define CICS group • Define CICS transaction names • Define a program to CICS 	<ul style="list-style-type: none"> • Chapter 4, “(Optional) Application Deployment Manager,” on page 65 • “(Optional) CICS bidirectional language support” on page 84

Table 4. Administrators needed for optional tasks (continued)

Administrator	Task	Information
DB2®	Define a DB2 stored procedure	“(Optional) DB2 stored procedure” on page 81
WLM	<ul style="list-style-type: none"> Assign goals to a DB2 stored procedure Assign TSO-like goals to an APPC transaction 	<ul style="list-style-type: none"> “(Optional) DB2 stored procedure” on page 81 Chapter 12, “WLM considerations,” on page 187
APPC	Define an APPC transaction	“(Optional) APPC transaction for the TSO Commands service” on page 95

Pre-configuration considerations

Workload management

Unlike traditional z/OS applications, Developer for System z is not a monolithic application that can be identified easily to Workload Manager (WLM). Developer for System z consists of several components that interact to give the client access to the host services and data. Refer to Chapter 12, “WLM considerations,” on page 187 to plan your WLM configuration accordingly.

Resource usage and system limits

When in use, Developer for System z will use a variable number of system resources like address spaces and z/OS UNIX processes and threads. The availability of these resources is limited by various system definitions. Refer to Chapter 13, “Tuning considerations,” on page 195 to estimate the usage of key resources, so you can plan your system configuration accordingly.

Required configuration of requisite products

Consult your MVS system programmer, security administrator and TCP/IP administrator to check if the requisite products and software are installed, tested, and working. Some requisite customization tasks that are easily overlooked are listed below:

- All Developer for System z users must have READ and EXECUTE access to the Java directories.
- All Developer for System z users must have READ, WRITE and EXECUTE access to the /tmp/ directory.
- Remote (host based) actions for z/OS UNIX subprojects require that z/OS UNIX version of REXEC or SSH is active on the host.

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).

Note: A large region size is required for the user ID that executes the Installation Verification Programs (IVPs), because functions requiring a lot of memory (such as Java) will be executed. You should set the region size to 131072 kilobytes (128 megabytes) or higher.

- An OMVS segment defined to the security system (for example, RACF®), both for the user ID and its default group.
 - The HOME field must refer to a home directory allocated for the user (with READ, WRITE and EXECUTE access).
 - 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
THREDSMAX= NONE
MMAPAREAM= 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 and files, default /usr/lpp/rdz/*, /etc/rdz/*, and /var/rdz/*.
- READ, WRITE, and EXECUTE access to the Developer for System z WORKAREA directory, default /var/rdz/WORKAREA.
- READ access to the Developer for System z installation data sets, default FEK.SFEK*.

Server considerations

Developer for System z consists of 3 permanently active servers, which can be started tasks or user jobs. These servers provide the requested services themselves, or start other servers (as z/OS UNIX threads or user jobs) to provide the service.

- JES Job Monitor (JMON) provides all JES-related services.
- Lock Daemon (LOCKD) provides tracking services for data set locks.
- Remote Systems Explorer (RSE) provides core services such as connecting the client to the host and starting other servers for specific services. RSE consists of 2 logical entities:
 - RSE daemon (RSED), which manages connection setup and which is responsible for running in single server mode.
 - RSE server, which handles individual client request.

JES Job Monitor (JMON) provides all JES related services.

- The security mechanisms used by JES Job Monitor rely on the data sets it resides in being secure. This implies that only trusted system administrators should be able to update the libraries and configuration files.

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

- Since version 7.5, RSE daemon is no longer an INETD managed process but a started task.
- Since version 7.5, RSE server uses a single server model whereas with previous versions, each client-host connection had a private RSE server.
- Different levels of communication security are supported by RSE:
 - External (client-host) communication can be limited to specified ports. This feature is disabled by default.
 - External (client-host) communication can be encrypted using SSL. This feature is disabled by default.
 - Port Of Entry (POE) checking can be used to allow access only to trusted TCP/IP addresses. This feature is disabled by default.
- RSE also supports multiple client authentication methods:
 - User ID and password
 - User ID and one-time password
 - X.509 certificate
- The security mechanisms used by RSE rely on the file system it resides in being secure. This implies that only trusted system administrators should be able to update the libraries and configuration files.

As documented in “TCP/IP ports” on page 150, certain host services, and thus their ports, must be available for the client to connect to, and must be defined to your firewall protecting the host. All other ports used by Developer for System z have host-only traffic. Listed below are the ports needed for a basic Developer for System z setup.

- RSE daemon for client-host communication setup (using the tcp protocol), default port 4035.
- RSE server for client-host communication (using the tcp protocol). By default, any available port is used, but this can be limited to a specified range.

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

Configuration method

Beginning with version 7.6.1, Developer for System z provides an alternative method, using an ISPF panel application, to configure the host side of the product. This gives you a choice of the following methods:

- Using the ISPF panel application. This guides you through the required customization steps and selected optional customization steps. For more information, refer to the *Host Configuration Utility* whitepaper, available at the Developer for System z internet library, <http://www-306.ibm.com/software/awdtools/rdz/library/>.
- Using the *Host Configuration Quick Start Guide*. This guides you through the required customization steps. The scope of this guide is limited to a basic setup.
- Using the *Host Configuration Guide*. This guides you through the required customization steps and all optional customization steps. All configurable options are covered in this guide, including some non-default scenarios

Predeployment considerations

Developer for System z supports 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, then this file must replace its counterpart in the lists below.

Note: The list below does not cover the deployment needs of the pre- and corequisite software.

- FEK.SFEKAUTH(*)
- FEK.SFEKLOAD(*)
- FEK.SFEKPROC(*)
- FEK.#CUST.PARMLIB(*)
- FEK.#CUST.PROCLIB(*)
- /usr/lpp/rdz/*
- /etc/rdz/*
- /var/rdz/* (directory structure only)
- optional parts:
 - FEK.SFEKLPA(*)
 - FEK.#CUST.CNTL(*)
 - definitions, data sets, files, and directories resulting from customization jobs in FEK.#CUST.JCL

Notes:

1. FEK and /usr/lpp/rdz are the high level qualifier and path used during the installation of the product. FEK.#CUST, /etc/rdz and /var/rdz are the default locations used during the customization of the product (see “Customization setup” on page 13 for more information).
2. You should install Developer for System z in a private file system (HFS or zFS) to easily deploying the z/OS UNIX parts of the product.
3. If you can not use a private file system, you should use an archiving tool such as the z/OS UNIX tar command to transport the z/OS UNIX directories from system to system. This to preserve the attributes (such as program control) for the Developer for System z files and directories.

Refer to *UNIX System Services Command Reference* (SA22-7802) for more information on the following sample commands to archive and restore the Developer for System z installation directory.

- Archive: `cd /SYS1/usr/lpp/rdz; tar -cSf /u/userid/rdz.tar`
- Restore: `cd /SYS2/usr/lpp/rdz; tar -xSf /u/userid/rdz.tar`

Client checklist

Users of the Developer for System z client must know the result of certain host customizations, such as TCP/IP port numbers, for the client to work properly. Use these checklists to gather the information needed.

The checklist in Table 5 on page 12 lists the required results of mandatory customization steps. Table 6 on page 12 list the required results of optional customization steps.

Table 5. Client checklist - mandatory parts

Customization	Value
JES Job Monitor server port number (default 6715): See SERV_PORT in "FEJJCNFG, JES Job Monitor configuration file" on page 24.	
RSE daemon TCP/IP port number (default 4035): See "RSE daemon" on page 19.	

Table 6. Client checklist - optional parts

Customization	Value
Location of the ELAXF* procedures if they are not in a system procedure library: See note on JCLLIB in "ELAXF* remote build procedures" on page 22.	
Procedure or step names of the ELAXF* procedures if they were changed: See note on changing them in "ELAXF* remote build procedures" on page 22.	
DB2 stored procedure name (default ELAXMSAM): See information on DB2 stored procedures in Chapter 17, "Running multiple instances," on page 249.	
Location of the DB2 stored procedure if it is not in a system procedure library: See "(Optional) DB2 stored procedure" on page 81.	
(corequisite) TN3270 port number for Host Connect Emulator (default 23). See Chapter 10, "Security considerations," on page 147	
(corequisite) REXEC or SSH port number (default 512 or 22, respectively): See "(Optional) Using REXEC (or SSH)" on page 93.	
Location of the server.zseries file if the REXEC/SSH connection method is used (default /etc/rdz). See "(Optional) Using REXEC (or SSH)" on page 93.	
Location of the CRA#ASLM JCL for CARMA SCLM RAM data set allocations (default FEK.#CUST.JCL): See note on CRA#ASLM in "Activating the SCLM RAM" on page 55.	

Chapter 2. Basic customization

The customization steps below are for a basic Developer for System z setup. Refer to the chapters about the optional components for their customization requirements.

Requirements and checklist

You will need the assistance of a security administrator and a TCP/IP administrator to complete this customization task, which requires the following resources and special customization tasks:

- APF authorized data set
- Various PARMLIB updates
- Various security software updates
- Various TCP/IP ports for internal and client-host communication

In order to verify the installation and to start using Developer for System z at your site, you must perform the following tasks. Unless otherwise indicated, all tasks are mandatory.

1. Create customizable copies of samples and create the work environment for Developer for system z. For details, see “Customization setup.”
2. Update z/OS UNIX system limits, start started tasks, define APF authorized and LINKLIST data sets and optionally LPA data sets. For details, see “PARMLIB changes” on page 14.
3. Create started task procedures and compile/link procedures. For details, see “PROCLIB changes” on page 19.
4. Update security definitions. For details, see “Security definitions” on page 23. You must also be aware of and understand how PassTickets are used to establish thread security. See “Using PassTickets” on page 152 for details.
5. Customize Developer for System z configuration files. For details, see:
 - “FEJCNFG, JES Job Monitor configuration file” on page 24
 - “rsed.envvars, RSE configuration file” on page 28
 - “ISPF.conf, ISPF’s TSO/ISPF Client Gateway configuration file” on page 42

Customization setup

Developer for System z comes with several sample configuration files and sample JCL. To avoid overwriting your customizations when applying maintenance, you should copy all these members and z/OS UNIX files to a different location and to customize the copy.

Some functions of Developer for System z also require the existence of certain directories in z/OS UNIX, which must be created during the customization of the product. To ease the installation effort, a sample job, FEKSETUP, is provided to create the copies and the required directories.

Customize and submit sample member FEKSETUP in data set FEK.SFEKSAMP to create customizable copies of configuration files and configuration JCL, and to create required z/OS UNIX directories. The required customization steps are described within the member.

This job performs the following tasks:

- Create FEK.#CUST.PARMLIB and populate it with sample configuration files.
- Create FEK.#CUST.PROCLIB and populate it with sample SYS1.PROCLIB members.
- Create FEK.#CUST.JCL and populate it with sample configuration JCL.
- Create FEK.#CUST.CNTL and populate it with sample server startup scripts.
- Create FEK.#CUST.ASM and populate it with sample assembler source code.
- Create FEK.#CUST.COBOL and populate it with sample COBOL source code.
- Create /etc/rdz/* and populate it with sample configuration files.
- Create /var/rdz/* as work directories for various Developer for System z functions.

Notes:

1. The configuration steps in this publication use the member/file locations created by the FEKSETUP job, unless noted otherwise. The original samples, which should not be updated, can be found in FEK.SFEKSAMP and /usr/lpp/rdz/samples/.
2. If you want to keep all Developer for System z z/OS UNIX files in the same file system (HFS or zFS), but also want the configuration files placed in /etc/rdz, you can use symbolic links to solve this problem. The following sample z/OS UNIX commands create a new directory in the existing file system (/usr/lpp/rdz/cust) and define a symbolic link (/etc/rdz) to it:

```
mkdir /usr/lpp/rdz/cust  
ln -s /usr/lpp/rdz/cust /etc/rdz
```

PARMLIB changes

Refer to *MVS Initialization and Tuning Reference* (SA22-7592) for more information on the PARMLIB definitions listed below. Refer to *MVS System Commands* (SA22-7627) for more information on the sample console commands.

Set z/OS UNIX limits in BPXPRMxx

Remote Systems Explorer (RSE), which provides core services such as connecting the client to the host, is a z/OS UNIX based process. Therefore it is important to set correct values for the z/OS UNIX system limits in BPXPRMxx, based upon the number of concurrently active Developer for System z users and their average workload.

Refer to Chapter 13, “Tuning considerations,” on page 195 for more information on different BPXPRMxx defined limits and their impact on Developer for System z.

MAXASSIZE specifies the maximum address space (process) region size. Set MAXASSIZE in SYS1.PARMLIB(BPXPRMxx) to 2G. This is the maximum value allowed. This is a system-wide limit, and thus active for all z/OS UNIX address spaces. If this is not what you want, then you can set the limit also just for Developer for System z in your security software, as described in “Define the Developer for System z started tasks” on page 169.

MAXTHREADS specifies the maximum number of active threads for a single process. Set MAXTHREADS in SYS1.PARMLIB(BPXPRMxx) to 1500 or higher. This is a system-wide limit, and thus active for all z/OS UNIX address spaces. If this is not what you want, then you can set the limit also just for Developer for System z in your security software, as described in “Define the Developer for System z started tasks” on page 169.

MAXTHREADTASKS specifies the maximum number of active MVS tasks for a single process. Set MAXTHREADTASKS in SYS1.PARMLIB(BPXPRMxx) to 1500 or higher. This is a system-wide limit, and thus active for all z/OS UNIX address spaces. If this is not what you want, then you can set the limit also just for Developer for System z in your security software, as described in “Define the Developer for System z started tasks” on page 169.

MAXPROCUSER specifies the maximum number of processes that a single z/OS UNIX user ID can have concurrently active. Set MAXPROCUSER in SYS1.PARMLIB(BPXPRMxx) to 50 or higher. This setting is intended to be a system-wide limit, as it should be active for each client using Developer for System z.

These values can be checked and set dynamically (until the next IPL) with the following console commands:

- DISPLAY OMVS,0
- SETOMVS MAXASSIZE=2G
- SETOMVS MAXTHREADS=1500
- SETOMVS MAXTHREADTASKS=1500
- SETOMVS MAXPROCUSER=50

Notes:

1. Refer to “Address Space size” on page 141 for more information on other locations where address space sizes can be set or limited.
2. The MAXPROCUSER value used above is based upon users having a unique z/OS UNIX user ID (UID). Increase this value if your users share the same UID.
3. Ensure that other BPXPRMxx values, such as those for MAXPROCSYS and MAXUIDS, are sufficient to handle the expected amount of concurrently active Developer for System z users. Refer to Chapter 13, “Tuning considerations,” on page 195 for more details.

Add started tasks to COMMNDxx

Add start commands for the Developer for System z RSED, LOCKD, and JMON servers to SYS1.PARMLIB(COMMANDxx) to start them automatically at next system IPL.

Once the servers are defined and configured, they can be started dynamically (until the next IPL) with the following console commands:

- S RSED
- S LOCKD
- S JMON

Note: The lock daemon should be started before Developer for System z users log on to the RSE daemon. This is so the lock daemon can track the data set lock requests by these users. Therefore you should start the lock daemon at system startup.

LPA definitions in LPALSTxx

The (optional) Common Access Repository Manager (CARMA) service supports alternative server startup methods that do not require the usage of a JES initiator. The most flexible of these alternatives requires that module CRASTART in the FEK.SFEKLPA load library is in the Link Pack Area (LPA).

LPA data sets are defined in SYS1.PARMLIB(LPALSTxx).

LPA definitions can be set dynamically (until the next IPL) with the following console commands:

- SETPROG LPA,ADD,DSN=FEK.SFEKLPA

APF authorizations in PROGxx

In order for JES Job Monitor to access JES spool files, module FEJJMON in the FEK.SFEKAUTH load library and the Language Environment® (LE) runtime libraries (CEE.SCEERUN*) must be APF authorized.

In order for the (optional) SCLM Developer Toolkit service to work, module BWBTSOW in the FEK.SFEKAUTH load library and the REXX runtime library (REXX.*.SEAGLPA) must be APF authorized.

In order for ISPF to create the TSO/ISPF Client Gateway, module ISPZTS0 in SYS1.LINKLIB must be APF authorized. The TSO/ISPF Client Gateway is used by Developer for System z's TSO Commands service, SCLM Developer Toolkit and optionally CARMA.

APF authorizations are defined in SYS1.PARMLIB(PROGxx), if your site followed IBM recommendations.

APF authorizations can be set dynamically (until the next IPL) with the following console commands, where volser is the volume on which the data set resides if it is not SMS managed:

- SETPROG APF,ADD,DSN=FEK.SFEKAUTH,SMS
- SETPROG APF,ADD,DSN=CEE.SCEERUN,VOL=volser
- SETPROG APF,ADD,DSN=CEE.SCEERUN2,VOL=volser
- SETPROG APF,ADD,DSN=REXX.V1R4M0.SEAGLPA,VOL=volser
- SETPROG APF,ADD,DSN=SYS1.LINKLIB,VOL=volser

Notes:

1. When you use the Alternate Library for REXX product package, the default REXX runtime library name is REXX.*.SEAGALT, instead of REXX.*.SEAGLPA as used in the preceding sample.
2. LPA libraries, such as REXX.*.SEAGLPA, are automatically APF authorized when located in LPA, and thus do not require explicit definitions.
3. Some of the corequisite products, such as IBM Debug Tool, also require APF authorization. Refer to the related product customization guides for more information on this.

LINKLIST definitions in PROGxx

LINKLIST definitions for Developer for System z can be grouped in 3 categories:

- Developer for System z load libraries needed for Developer for System z functions. These definitions are described in this section.
- Requisite load libraries needed for Developer for System z functions. These definitions are described in "Requisite LINKLIST and LPA definitions" on page 17.
- Developer for System z load libraries needed by other products. These definitions are described in "LINKLIST definitions for other products" on page 18.

In order for the (optional) SCLM Developer Toolkit service to work, all BWB* modules in the FEK.SFEKAUTH and FEK.SFEKLOAD load libraries must be made available either through STEPLIB or LINKLIST.

If you opt to use STEPLIB, you must define the libraries not available through LINKLIST in the STEPLIB directive of rsed.envvars, the RSE configuration file. Be aware, however, that:

- using STEPLIB in z/OS UNIX has a negative performance impact.
- If one STEPLIB library is APF authorized, then all must be authorized. Libraries lose their APF authorization when they are mixed with non-authorized libraries in STEPLIB.

LINKLIST data sets are defined in SYS1.PARMLIB(PROGxx), if your site followed IBM recommendations.

The required definitions will look like the following, where listname is the name of the LINKLIST set that will be activated, and volser is the volume on which the data set resides if it is not cataloged in the master catalog:

- LNKST ADD NAME(listname) DSN=FEK.SFEKAUTH VOLUME(volser)
- LNKST ADD NAME(listname) DSN=FEK.SFEKLOAD

LINKLIST definitions can be created dynamically (until the next IPL) with the following group of console commands, where listname is the name of the current LINKLIST set, and volser is the volume on which the data set resides if it is not cataloged in the master catalog:

1. LNKST DEFINE,NAME=LLTMP,COPYFROM=CURRENT
2. LNKST ADD NAME=LLTMP,DSN=FEK.SFEKAUTH,VOL=volser
3. LNKST ADD NAME=LLTMP,DSN=FEK.SFEKLOAD
4. LNKST ACTIVATE,NAME=LLTMP
5. LNKST UNDEFINE,NAME=listname
6. LNKST UPDATE,JOB=*

Requisite LINKLIST and LPA definitions

Remote Systems Explorer (RSE) is a z/OS UNIX process that requires access to MVS load libraries. The following (prerequisite) libraries must be made available, either through STEPLIB or LINKLIST/LPALIB:

- System load library
 - SYS1.LINKLIB
- Language Environment runtime
 - CEE.SCEERUN
 - CEE.SCEERUN2
- C++'s DLL class library
 - CBC.SCLBDLL
- ISPF's TSO/ISPF Client Gateway
 - ISP.SISPLoad
 - ISP.SISPLPA

The following additional libraries must be made available, either through STEPLIB or LINKLIST/LPALIB, to support the use of optional services. This list does not include data sets that are specific to a product that Developer for System z interacts with, such as IBM Debug Tool:

- REXX runtime library (for SCLM Developer Toolkit)
 - REXX.*.SEAGLPA
- System load library (for SSL encryption)
 - SYS1.SIEALNKE
- TCP/IP load library (when using APPC for the TSO Commands service)
 - TCP/IP.SEZALOAD

Notes:

1. When you use the Alternate Library for REXX product package, the default REXX runtime library name is REXX.*.SEAGALT, instead of REXX.*.SEAGLPA as used in the preceding sample.
2. Libraries that are designed for LPA placement, such as REXX.*.SEAGLPA, might require additional program control and/or APF authorizations if they are accessed through LINKLIST or STEPLIB.
3. Some of the corequisite products, such as IBM Debug Tool, also require STEPLIB or LINKLIST/LPALIB definitions. Refer to the related product customization guides for more information on this.
4. If CEE.SCEELKED is in LINKLIST or STEPLIB, TCP/IP.SEZALOAD must be placed before CEE.SCEELKED. Failure to do so will result in a 0C1 system abend for the TCP/IP REXX socket calls.

LINKLIST data sets are defined in SYS1.PARMLIB(PROGxx), if your site followed IBM recommendations. LPA data sets are defined in SYS1.PARMLIB(LPALSTxx).

If you opt to use STEPLIB, you must define the libraries not available through LINKLIST/LPALIB in the STEPLIB directive of rsed.envvars, the RSE configuration file. Be aware, however, that:

- Using STEPLIB in z/OS UNIX has a negative performance impact.
- If one STEPLIB library is APF authorized, then all must be authorized. Libraries lose their APF authorization when they are mixed with non-authorized libraries in STEPLIB.
- Libraries added to the STEPLIB DD in a JCL are not propagated to the z/OS UNIX processes started by the JCL.

LINKLIST definitions for other products

The Developer for System z client has a code generation component called Enterprise Service Tools (EST). In order for the generated code to issue diagnostic error messages, all IRZ* and IIRZ* modules in the FEK.SFEKLOAD load library must be made available either through STEPLIB or LINKLIST.

LINKLIST data sets are defined in SYS1.PARMLIB(PROGxx), if your site followed IBM recommendations.

If you opt to use STEPLIB, you must define the libraries not available through LINKLIST in the STEPLIB directive of the task that executes the code (IMS™ or batch job). However, be aware of the following:

- If one STEPLIB library is APF authorized, then all must be authorized. Libraries lose their APF authorization when they are mixed with non-authorized libraries in STEPLIB.

PROCLIB changes

The started task and remote build procedures listed below must reside in a system procedure library defined to your JES subsystem. In the instructions below, the IBM default procedure library, SYS1.PROCLIB, is used.

JES Job Monitor

Customize the sample started task member FEK.#CUST.PROCLIB(JMON), as described within the member, and copy it to SYS1.PROCLIB. As shown in the code sample below, you have to provide the following:

- The high level qualifier of the (authorized) load library, default FEK
- The JES Job Monitor configuration file, default FEK.#CUST.PARMLIB(FEJJCNFG)

```

/*
/* JES JOB MONITOR
/*
//JMON      PROC PRM=,                * PRM='-TV' TO START TRACING
//          LEPRM='RPTOPTS(ON)',
//          HLQ=FEK,
//          CFG=FEK.#CUST.PARMLIB(FEJJCNFG)
/*
//JMON      EXEC PGM=FEJJMON,REGION=0M,TIME=NOLIMIT,
//          PARM=('&LEPRM,ENVAR("_CEE_ENVFILE_S=DD:ENVIRON")/&PRM')
//STEPLIB   DD DISP=SHR,DSN=&HLQ..SF&HLQ
//ENVIRON   DD DISP=SHR,DSN=&CFG
//SYSPRINT  DD SYSOUT=*
//SYSOUT    DD SYSOUT=*
//          PEND
/*

```

Figure 1. JMON - JES Job Monitor started task

Notes:

1. Refer to Chapter 8, “Operator commands,” on page 115 for more information on the startup parameters.
2. The sample JCL is initially shipped as FEK.SFEKSAMP(FEJJJCL) and is renamed to FEK.#CUST.PROCLIB(JMON) in “Customization setup” on page 13.
3. Tracing can also be controlled by console commands, as described in Chapter 8, “Operator commands,” on page 115.
4. This task must be assigned to SYSSTC or equivalent goal in Workload Manager (WLM).
5. The LE environment variable _CEE_ENVFILE_S requires z/OS 1.8 or higher. The variable can be substituted with _CEE_ENVFILE on older z/OS levels, but due to a bug in the C runtime, the TZ variable in the JES Job Monitor configuration file (FEJJCNFG) might not be interpreted correctly.

RSE daemon

Customize the sample started task member FEK.#CUST.PROCLIB(RSED), as described within the member, and copy it to SYS1.PROCLIB. As shown in the code sample below, you have to provide the following:

- The RSE daemon port, default 4035.

- The home directory where Developer for System z is installed, default /usr/lpp/rdz.
- The location of the configuration files, default /etc/rdz

```

/*
/* RSE DAEMON
/*
//RSED    PROC IVP='',          * 'IVP' to do an IVP test
//        PORT=4035,
//        HOME='/usr/lpp/rdz',
//        CNFG='/etc/rdz'
/*
//RSE     EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT,
//        PARM='PGM &HOME/bin/rsed.sh &IVP &PORT &CNFG'
//STDERR  DD SYSOUT=*
//STDOUT  DD SYSOUT=*
//        PEND
/*

```

Figure 2. RSED - RSE daemon started task

Note:

- Refer to Chapter 8, “Operator commands,” on page 115 for more information on the startup parameters.
- The sample JCL is initially shipped as FEK.SFEKSAMP(FEKRSED) and is renamed to FEK.#CUST.PROCLIB(RSED) in “Customization setup” on page 13.
- Limit the length of the job name to 7 characters or less. The **modify** and **stop** operator commands will fail with message "IEE342I MODIFY REJECTED-TASK BUSY" if an 8 character name is used. This behavior is caused by the z/OS UNIX design for child processes.
- This task, and the child processes it creates, must be assigned to SYSSTC or equivalent goal in Workload Manager (WLM). The child processes have the same name as the parent task, RSED, appended with a random 1-digit number, for example RSED8.

Lock daemon

Customize the sample started task member FEK.#CUST.PROCLIB(LOCKD), as described within the member, and copy it to SYS1.PROCLIB. As shown in the code sample below, you have to provide the following:

- The home directory where Developer for System z is installed, default /usr/lpp/rdz.
- The location of the configuration files, default /etc/rdz.
- The initial log detail level, default 1.

```

/*
/* RSE LOCK DAEMON
/*
//LOCKD    PROC HOME='/usr/lpp/rdz',
//          CNFG='/etc/rdz',
//          LOG=1
/*
//LOCKD    EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT,
//          PARM=PGM &HOME./bin/lockd.sh &CNFG &LOG'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//          PEND
/*

```

Figure 3. LOCKD - Lock daemon started task

Notes:

1. Refer to Chapter 8, “Operator commands,” on page 115 for more information on the startup parameters.
2. The sample JCL is initially shipped as FEK.SFEKSAMP(FEKLOCKD) and is renamed to FEK.#CUST.PROCLIB(LOCKD) in “Customization setup” on page 13.
3. This task must be assigned to SYSSTC or equivalent goal in Workload Manager (WLM).

JCL limitations for the PARM variable

The maximum length for the PARM variable is 100 characters, which might cause problems if you use custom directory names. To bypass this problem, you can either:

- Use symbolic links

Symbolic links can be used as shorthand for a long directory name. The following sample z/OS UNIX command defines a symbolic link (/usr/lpp/rdz) to another directory (/long/directory/name/usr/lpp/rdz).

```
ln -s /long/directory/name/usr/lpp/rdz /usr/lpp/rdz
```

- Use STDIN

When the PARM field is empty, **BPXBATSL** will start a z/OS UNIX shell and execute the shell script that is provided by STDIN. Note that STDIN must be a z/OS UNIX file (allocated as ORDONLY) and that using STDIN disables the usage of PROC variables for the port etc. Also note that the shell will execute the shell logon scripts /etc/profile and \$HOME/.profile.

To use this method, you must first update the startup JCL to match something like the following sample:

```

/*
/* RSE DAEMON - USING STDIN
/*
//RSED     PROC CNFG='/etc/rdz'
/*
//RSE      EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//STDIN    DD PATHOPTS=(ORDONLY),PATH='&CNFG./rsed.stdin.sh'
//STDENV   DD PATHOPTS=(ORDONLY),PATH='&CNFG./rsed.envvars'
//          PEND
/*

```

Figure 4. RSED - alternate RSE daemon startup

Second, you must create the shell script (/etc/rdz/rsed.stdin.sh in this example) that will start the RSE daemon. The content of this script will look like the following sample:

```
/long/directory/name/usr/lpp/rdz/bin/rsed.sh 4035 /etc/rdz
```

Figure 5. rsed.stdin.sh - alternate RSE daemon startup

Note: You should allocate rsed.envvars to STDENV in the daemon startup JCL, as it defines some z/OS UNIX directives that help save system resources when using this startup method.

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, COBOL, PL/I, Assembler, and C/C++ programs. These procedures allow installations to apply their own standards, and ensure that developers use the same procedures with the same compiler options and compiler levels.

The sample procedures and their function 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.
ELAXFCP1	Sample procedure for COBOL compiles with SCM preprocessor statements (-INC and ++INCLUDE).
ELAXFDCL	Sample procedure for running a program in TSO mode.
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.

Table 7. Sample ELAXF* procedures (continued)

Member	Purpose
ELAXFPP1	Sample procedure for PL/I compiles with SCM preprocessor statements (-INC and ++INCLUDE).
ELAXFTSO	Sample procedure for running/debugging generated DB2 code in TSO mode.
ELAXFUOP	Sample procedure for generating the UOPT step when building programs that run in CICS or IMS subsystems.

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

Customize the sample build procedure members, FEK.#CUST.PROCLIB(ELAXF*), as described within the members, and copy them to SYS1.PROCLIB. You have to provide the correct high level qualifiers for different product libraries, as described in Table 8.

Table 8. ELAXF* high level qualifier checklist

Product	Default HLQ	Value
Developer for System z	FEK	
CICS	CICSTS32.CICS	
DB2	DSN910	
IMS	IMS	
COBOL	IGY.V4R1M0	
PL/I	IBMZ.V3R8M0	
C/C++	CBC	
LE	CEE	
system LINKLIB	SYS1	
system MACLIB	SYS1	

If the ELAXF* procedures cannot be copied into a system procedure library, ask the Developer for System z users to add a JCLLIB card (right after the JOB card) to the job properties on the client.

```
//MYJOB    JOB <job parameters>
//PROCS    JCLLIB ORDER=(FEK.#CUST.PROCLIB)
```

Security definitions

Customize and submit sample member FEKRACF in data set FEK.#CUST.JCL to create the security definitions for Developer for System z. The user submitting this job must have security administrator privileges, such as being RACF SPECIAL.

Note:

- For those sites that use CA ACF2™ for z/OS, please refer to the following link, <https://support.ca.com/irj/portal/kbtech?ipLogNrow=0>

&docid=492389&searchID=TEC492389, for details on the security commands necessary to properly configure Developer for System z.

- For those sites that use CA Top Secret® for z/OS, please refer to your product page on the CA support site (<https://support.ca.com>) and check for the related Developer for System z Knowledge Document. This Knowledge Document has details on the security commands necessary to properly configure Developer for System z.

The following list of mandatory security-related definitions for Developer for System z are discussed in detail in Chapter 10, “Security considerations,” on page 147. This chapter also discusses general security-related aspects of Developer for System z, including security aspects of requisite products that are not covered by the sample FEKRACF job.

- Activate security settings and classes
- Define an OMVS segment for Developer for System z users
- Define data set profiles
- Define the JMON, RSED, and LOCKD started tasks
- Define JES command security
- Define RSE as a secure z/OS UNIX server
- Define MVS program controlled libraries for RSE
- Define application security for RSE
- Define PassTicket support for RSE
- Define z/OS UNIX program controlled files for RSE

Note: The sample FEKRACF job holds more than just RACF commands. The last step of the security definitions consists of making a z/OS UNIX file program controlled. Depending on the policies at your site, this might be a task for the system programmer and not the security administrator.

Attention: The client connection request will fail if application security and PassTickets are not set up correctly.

FEJJCNFG, JES Job Monitor configuration file

JES Job Monitor (JMON) provides all JES-related services. The behavior of JES Job Monitor can be controlled with the definitions in FEJJCNFG.

FEJJCNFG is located in FEK.#CUST.PARMLIB, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP (FEKSETUP). See “Customization setup” on page 13 for more details.

Customize the sample JES Job Monitor configuration member FEJJCNFG, as shown in the following sample. Comment lines start with a pound sign (#), when using a US code page. Data lines can only have a directive and its assigned value, comments are not allowed on the same line.

Note: The JMON started task must be restarted to pick up any changes you make.

```

SERV_PORT=6715
TZ=EST5EDT
#_BPXK_SETIBMOPT_TRANSPORT=TCPIP
#APPLID=FEKAPPL
#AUTHMETHOD=SAF
#CODEPAGE=UTF-8
#CONCHAR=$
#CONSOLE_NAME=JMON
#GEN_CONSOLE_NAME=OFF
#HOST_CODEPAGE=IBM-1047
#LIMIT_COMMANDS=NOLIMIT
#LIMIT_VIEW=USERID
#LISTEN_QUEUE_LENGTH=5
#MAX_DATASETS=32
#MAX_THREADS=200
#TIMEOUT=3600
#TIMEOUT_INTERVAL=1200
#SUBMITMETHOD=TSO
#TSO_TEMPLATE=FEK.#CUST.CNTL(FEJTS0)

```

Figure 6. FEJJCNFG, JES Job Monitor configuration file

SERV_PORT

The port number for JES Job Monitor host server. The default port is 6715. Change as desired, however, BOTH the server and the Developer for System z clients must be configured with the same port number. If you change the server port number, all clients must also change the JES Job Monitor port for this system in the 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**.
- When using a version 7.1 or higher client, all communication on this port is confined to your z/OS host machine.

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 Command Reference* (SA22-7802).

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

_BPXK_SETIBMOPT_TRANSPORT

Specifies the name of the TCPIP stack to be used. The default is TCPIP. Uncomment and change to the requested TCPIP stack name, as defined in the TCPIPJOBNAME statement in the related TCPIP.DATA.

Note:

- Coding a SYSTCPD DD statement in the server JCL does not set the requested stack affinity.
- When this directive is not active, JES Job Monitor binds to every available stack on the system (BIND INADDRANY).

APPLID

Specifies the application identifier used for identifying JES Job Monitor to your security software. The default is FEKAPPL. Uncomment and change to the desired application ID.

Note: This value must match the application ID set for RSE in the `rsed.envvars` configuration file. If these values differ, RSE cannot connect the client to JES Job Monitor.

AUTHMETHOD

The default is `SAF`, which means that the System Authorization Facility (SAF) security interface is used. Do not change unless directed to do so by the IBM support center.

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 uncomment the directive and change `UTF-8` to match the workstation's codepage if you have difficulty with NLS characters, such as the currency symbol.

CONCHAR

Specifies the JES console command character. `CONCHAR` defaults to `CONCHAR=$` for JES2, or `CONCHAR=*` for JES3. Uncomment and change to the requested command character.

CONSOLE_NAME

Specifies the name of the EMCS console used for issuing commands against jobs (Hold, Release, Cancel, and Purge). The default is `JMON`. Uncomment and change to the desired console name, using the guidelines below.

- `CONSOLE_NAME` must be either a console name consisting of 2 to 8 alphanumeric characters, or `'&SYSUID'` (without quotes).
- If a console name is specified, a single console by that name is used for all users. If the console by that name happens to be in use, then the command issued by the client will fail.
- If `&SYSUID` is specified, the client user ID is used as the console name. Thus a different console is used for each user. If the console by that name happens to be in use (for example, the user is using the `SDSF ULOG`), then the command issued by the client might fail, depending on the `GEN_CONSOLE_NAME` setting.

No matter which console name is used, the user ID of the client requesting the command is used as the LU of the console, leaving a trace in syslog messages `IEA630I` and `IEA631I`.

```
IEA630I OPERATOR console NOW ACTIVE,   SYSTEM=sysid, LU=id
IEA631I OPERATOR console NOW INACTIVE, SYSTEM=sysid, LU=id
```

GEN_CONSOLE_NAME

Enables or disables automatic generating of alternative console names. The default is `OFF`. Uncomment and change to `ON` to enable alternative console names.

This directive is only used when `CONSOLE_NAME` equals `&SYSUID` and the user ID is not available as console name.

If `GEN_CONSOLE_NAME=ON`, an alternative console name is generated by appending a single numeric digit to the user ID. The digits 0 through 9 are attempted. If no available console is found, the command issued by the client fails.

If `GEN_CONSOLE_NAME=OFF`, the command issued by the client fails.

Note: The only valid settings are `ON` and `OFF`.

HOST_CODEPAGE

The host codepage. The default is IBM-1047. Uncomment and change to match your host codepage.

From version 7.6.1 on, Developer for System z clients ignore the HOST_CODEPAGE value specified here and use the codepage specified locally in the properties of the “MVS Files” subsystem.

Note: Even for recent clients, JES Job Monitor will use the host codepage specified in HOST_CODEPAGE during initial client communication setup.

LIMIT_COMMANDS

Defines against which jobs the user can issue selected JES commands (Show JCL, Hold, Release, Cancel, and Purge). The default (LIMIT_COMMANDS=USERID) limits the commands to jobs owned by the user. Uncomment this directive and specify LIMITED or NOLIMIT to allow the user to issue commands against all spool files, if permitted by your security product.

Table 9. LIMIT_COMMANDS command permission matrix

LIMIT_COMMANDS	Job owner	
	User	Other
USERID (default)	Allowed	Not allowed
LIMITED	Allowed	Allowed only if explicitly permitted by security profiles
NOLIMIT	Allowed	Allowed if permitted by security profiles or when the JESSPOOL class is not active

Note: The only valid settings are USERID, LIMITED, and NOLIMIT.

LIMIT_VIEW

Defines what output the user can view. The default (LIMIT_VIEW=NOLIMIT) allows the user to view all JES output, if permitted by your security product. Uncomment this directive and specify USERID to limit the view to output owned by the user.

Note: The only valid settings are USERID and NOLIMIT.

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 maximum number of spooled output data sets that JES Job Monitor will return to the client (for example, SYSOUT, SYSPRINT, SYS00001, and so on). The default is 32. The maximum value is 2147483647.

MAX_THREADS

Maximum number of users that can be using one JES Job Monitor at a time. The default is 200. The maximum value is 2147483647. Increasing this number may require you to increase the size of the JES Job Monitor address space.

TIMEOUT

The length of time, in seconds, before a thread is killed due to lack of

interaction with the client. The default is 3600 (1 hour). The maximum value is 2147483647. TIMEOUT=0 disables the function.

TIMEOUT_INTERVAL

The number of seconds between timeout checks. The default is 1200. The maximum value is 2147483647.

SUBMITMETHOD=TSO

Submit jobs through TSO. The default (SUBMITMETHOD=JES) submits jobs directly into JES. Uncomment this directive and specify TSO to submit the job through TSO **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.
- If SUBMITMETHOD=TSO is specified, then TSO_TEMPLATE must also be defined.

TSO_TEMPLATE

Wrapper JCL for submitting the job through TSO. The default value is FEK.#CUST.CNTL(FEJTSO). This statement refers to 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 FEK.#CUST.CNTL(FEJTSO). Refer to this member for more information on the customization needed.
- TSO_TEMPLATE has no effect unless SUBMITMETHOD=TSO is also specified.

rsed.envvars, RSE configuration file

The RSE lock daemon and the RSE server processes (RSE daemon, RSE thread pool, and RSE server) use the definitions in rsed.envvars. Optional Developer for System z and third-party services can use this configuration file also to define environment variables for their use.

Remote Systems Explorer (RSE) provides core services such as connecting the client to the host and starting other servers for specific services. Lock daemon provides tracking services for data set locks.

rsed.envvars is located in /etc/rdz/, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details. You can edit the file with the TSO **OEDIT** command.

See the following sample rsed.envvars file, which must be customized to match your system environment. Comment lines start with a pound sign (#), when using a US code page. Data lines can only have a directive and its assigned value, comments are not allowed on the same line. Line continuations and spaces around the equal sign (=) are not supported.

Note: The RSED and LOCKD started tasks must be restarted to pick up any changes you make.


```

#=====
# (1) required definitions
JAVA_HOME=/usr/lpp/java/J5.0
RSE_HOME=/usr/lpp/rdz
_RSE_LOCKD_PORT=4036
_RSE_HOST_CODEPAGE=IBM-1047
TZ=EST5EDT
LANG=C
PATH=/bin:/usr/sbin
_CEE_DMPTARG=/tmp
STEPLIB=NONE
#STEPLIB=$STEPLIB:CEE.SCEERUN:CEE.SCEERUN2:CBC.SCLBDLL
_RSE_SAF_CLASS=/usr/include/java_classes/IRRRacf.jar
_RSE_JAVAOPTS=""
_RSE_JAVAOPTS="$ _RSE_JAVAOPTS -Xms1m -Xmx256m"
_RSE_JAVAOPTS="$ _RSE_JAVAOPTS -Ddaemon.log=/var/rdz/logs"
_RSE_JAVAOPTS="$ _RSE_JAVAOPTS -Duser.log=/var/rdz/logs"
_RSE_JAVAOPTS="$ _RSE_JAVAOPTS -DDSTORE_LOG_DIRECTORY="
# _RSE_JAVAOPTS="$ _RSE_JAVAOPTS -Dmaximum.clients=60"
# _RSE_JAVAOPTS="$ _RSE_JAVAOPTS -Dmaximum.threads=1000"
# _RSE_JAVAOPTS="$ _RSE_JAVAOPTS -Dminimum.threadpool.process=1"
# _RSE_JAVAOPTS="$ _RSE_JAVAOPTS -Dmaximum.threadpool.process=100"
# _RSE_JAVAOPTS="$ _RSE_JAVAOPTS -Dipv6=true"
# _RSE_JAVAOPTS="$ _RSE_JAVAOPTS -Dkeep.last.log=true"
# _RSE_JAVAOPTS="$ _RSE_JAVAOPTS -Denable.standard.log=true"
# _RSE_JAVAOPTS="$ _RSE_JAVAOPTS -Denable.port.of.entry=true"
# _RSE_JAVAOPTS="$ _RSE_JAVAOPTS -Denable.certificate.mapping=false"
# _RSE_JAVAOPTS="$ _RSE_JAVAOPTS -Denable.automount=true"
# _RSE_JAVAOPTS="$ _RSE_JAVAOPTS -Denable.audit.log=true"
# _RSE_JAVAOPTS="$ _RSE_JAVAOPTS -Daudit.cycle=30"
# _RSE_JAVAOPTS="$ _RSE_JAVAOPTS -Daudit.retention.period=0"
# _RSE_JAVAOPTS="$ _RSE_JAVAOPTS -Ddeny.nonzero.port=true"
# _RSE_JAVAOPTS="$ _RSE_JAVAOPTS -Dsingle.logon=false"
# _RSE_JAVAOPTS="$ _RSE_JAVAOPTS -Dprocess.cleanup.interval=0"
# _RSE_JAVAOPTS="$ _RSE_JAVAOPTS -DAPPLID=FEKAPPL"
# _RSE_JAVAOPTS="$ _RSE_JAVAOPTS -DDENY_PASSWORD_SAVE=true"
# _RSE_JAVAOPTS="$ _RSE_JAVAOPTS -Dhide_zos_unix=true"
# _RSE_JAVAOPTS="$ _RSE_JAVAOPTS -DDSTORE_IDLE_SHUTDOWN_TIMEOUT=3600000"
# _RSE_JAVAOPTS="$ _RSE_JAVAOPTS -DDSTORE_TRACING_ON=true"
# _RSE_JAVAOPTS="$ _RSE_JAVAOPTS -DDSTORE_MEMLOGGING_ON=true"
# _RSE_JAVAOPTS="$ _RSE_JAVAOPTS -DTSO_SERVER=APPC"
#=====
# (2) required definitions for TSO/ISPF Client Gateway
_CMDSERV_BASE_HOME=/usr/lpp/ispf
_CMDSERV_CONF_HOME=/etc/rdz
_CMDSERV_WORK_HOME=/var/rdz
#STEPLIB=$STEPLIB:ISP.SISPLPA:SYS1.LINKLIB
_RSE_CMDSERV_OPTS=""
# _RSE_CMDSERV_OPTS="$ _RSE_CMDSERV_OPTS&ISPPROF=&SYSUID..ISPPROF"
#=====
# (3) required definitions for SCLM Developer Toolkit
_SCLMDT_CONF_HOME=/var/rdz/sclmdt
#STEPLIB=$STEPLIB:FEK.SFEKAUTH:FEK.SFEKLOAD
# _SCLMDT_TRANTABLE=FEK.#CUST.LSTRANS.FILE
#ANT_HOME=/usr/lpp/Apache/Ant/apache-ant-1.7.1
#=====
# (4) optional definitions
# _RSE_PORTRANGE=8108-8118
# _BPXK_SETIBMOPT_TRANSPORT=TCPIP
# _FEKFSCMD_TP_NAME=_FEKFRSRV
# _FEKFSCMD_PARTNER_LU=_lu_name
#GSK_CRL_SECURITY_LEVEL=HIGH
#GSK_LDAP_SERVER=ldap_server_url
#GSK_LDAP_PORT=ldap_server_port
#GSK_LDAP_USER=ldap_userid
#GSK_LDAP_PASSWORD=ldap_server_password
#=====

```

Figure 7. *rsed.envvars* - RSE configuration file

```

# (5) do not change unless directed by IBM support center
_CEE_RUNOPTS="ALL31(ON) HEAP(32M,32K,ANYWHERE,KEEP,,) TRAP(ON)"
_BPX_SHAREAS=YES
_BPX_SPAWN_SCRIPT=YES
JAVA_PROPAGATE=NO
RSE_LIB=$RSE_HOME/lib
PATH=.:$JAVA_HOME/bin:$RSE_HOME/bin:$CMDSESV_BASE_HOME/bin:$PATH
LIBPATH=$JAVA_HOME/bin:$JAVA_HOME/bin/classic:$RSE_LIB:$RSE_LIB/icuc
LIBPATH=.:usr/lib:$LIBPATH
CLASSPATH=$RSE_LIB:$RSE_LIB/dstore_core.jar:$RSE_LIB/clientserver.jar
CLASSPATH=$CLASSPATH:$RSE_LIB/dstore_extra_server.jar
CLASSPATH=$CLASSPATH:$RSE_LIB/zosserver.jar
CLASSPATH=$CLASSPATH:$RSE_LIB/dstore_miners.jar
CLASSPATH=$CLASSPATH:$RSE_LIB/universalminers.jar:$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-core-2.3.2.jar
CLASSPATH=$CLASSPATH:$RSE_LIB/cdtparser.jar
CLASSPATH=$CLASSPATH:$RSE_LIB/wdzbidi.jar:$RSE_LIB/fmiExtensions.jar
CLASSPATH=$CLASSPATH:$RSE_SAF_CLASS
CLASSPATH=.:$CLASSPATH
_RSE_CMDSESV_OPTS="&SESSION=SPAWN$ RSE_CMDSESV_OPTS"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DISPF_OPTS='$_RSE_CMDSESV_OPTS'"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DA_PLUGIN_PATH=$RSE_LIB"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Xbootclasspath/p:$RSE_LIB/bidiTools.jar"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Dfile.encoding=$_RSE_HOST_CODEPAGE"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Dconsole.encoding=$_RSE_HOST_CODEPAGE"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DDSTORE_SPIRIT_ON=true"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DSPIRIT_EXPIRY_TIME=6"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DSPIRIT_INTERVAL_TIME=6"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Dcom.ibm.cacheLocalHost=true"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Duser.home=$HOME"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Dclient.username=$RSE_USER_ID"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Dlow.heap.usage.ratio=15"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Dmaximum.heap.usage.ratio=40"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DDSTORE_KEEPALIVE_ENABLED=true"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DDSTORE_KEEPALIVE_RESPONSE_TIMEOUT=30000"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DDSTORE_IO_SOCKET_READ_TIMEOUT=90000"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DRSECOMM_LOGFILE_MAX=0"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Dlock.daemon.port=$_RSE_LOCKD_PORT"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Dlock.daemon.cleanup.interval=1440"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -showversion"
_RSE_SERVER_CLASS=org.eclipse.dstore.core.server.Server
_RSE_DAEMON_CLASS=com.ibm.etools.zos.server.RseDaemon
_RSE_POOL_SERVER_CLASS=com.ibm.etools.zos.server.ThreadPoolProcess
_RSE_LOCKD_CLASS=com.ibm.ftt.rse.mvs.server.miners.MVSLockDaemon
_RSE_SERVER_TIMEOUT=120000
_SCLMDT_BASE_HOME=$RSE_HOME
_SCLMDT_WORK_HOME=$CMDSESV_WORK_HOME
CGI_DTWORK=$_SCLMDT_WORK_HOME
#=====
# (6) additional environment variables

```

Figure 8. (continued)

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

The following definitions are required:

JAVA_HOME

Java home directory. The default is `/usr/lpp/java/J5.0`. Change to match your Java installation.

RSE_HOME

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

_RSE_LOCKD_PORT

RSE lock daemon port number. The default is 4036. Can be changed if desired.

Note:

- Before selecting a port, verify that the port is available on your system with the TSO commands **NETSTAT** and **NETSTAT PORTL**.
- All communication on this port is confined to your z/OS host machine.

_RSE_HOST_CODEPAGE

The host codepage. The default is IBM-1047. Change to match your host codepage.

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 Command Reference* (SA22-7802).

LANG

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

PATH Command path. The default is `/bin:/usr/sbin:..`. Can be changed if desired.

_CEE_DMPTARG

Language Environment (LE) z/OS UNIX dump location used by the Java Virtual Machine (JVM). The default is `/tmp`.

STEPLIB

Access MVS data sets not in LINKLIST/LPALIB. The default is NONE.

You can bypass the need of having (prerequisite) libraries in LINKLIST/LPALIB by uncommenting and customizing one or more of the following STEPLIB directives. Refer to “PARMLIB changes” on page 14 for more information on the usage of the libraries listed below:

```
STEPLIB=$STEPLIB:CEE.SCEERUN:CEE.SCEERUN2:CBCLBDDL
STEPLIB=$STEPLIB:ISP.SISPLoad:ISP.SISPLPA:SYS1.LINKLIB
STEPLIB=$STEPLIB:FEK.SFEKAUTH:FEK.SFEKLOAD
```

Note:

- Using STEPLIB in z/OS UNIX has a negative performance impact.

- If one STEPLIB library is APF authorized, then all must be authorized. Libraries lose their APF authorization when they are mixed with non-authorized libraries in STEPLIB.
- Libraries that are designed for LPA placement might require additional program control and APF authorizations if they are accessed through LINKLIST or STEPLIB.
- Coding a STEPLIB DD statement in the server JCL does not set the requested STEPLIB concatenation.

RSE_SAF_CLASS

Specifies the Java interface to your security product. The default is /usr/include/java_classes/IRRRacf.jar. Change to match your security software setup.

Note: Since z/OS 1.10, /usr/include/java_classes/IRRRacf.jar is part of SAF, which ships with base z/OS, so it is available also to non-RACF customers.

RSE_JAVAOPTS

Additional RSE-specific Java options. . See “Defining extra Java startup parameters with _RSE_JAVAOPTS” on page 37 for more information on this definition.

Developer for System z uses ISPF’s TSO/ISPF Client Gateway by default for the TSO Commands service. An APPC transaction is used instead when the following _RSE_JAVAOPTS option is uncommented:

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

The following definitions are required if ISPF’s TSO/ISPF Client Gateway is used for the TSO Commands service, SCLM Developer Toolkit or CARMA.

_CMDSERV_BASE_HOME

Home directory for the ISPF code that provides the TSO/ISPF Client Gateway service. The default is /usr/lpp/ispf. Change to match your ISPF installation. This directive is only required when ISPF’s TSO/ISPF Client Gateway is used.

_CMDSERV_CONF_HOME

ISPF base configuration directory. The default is /etc/rdz. Change to match the location of ISPF.conf, the TSO/ISPF Client Gateway customization file. This directive is only required when ISPF’s TSO/ISPF Client Gateway is used.

_CMDSERV_WORK_HOME

ISPF base work directory. The default is /var/rdz. Change to match the location of the WORKAREA directory used by the TSO/ISPF Client Gateway. This directive is only required when ISPF’s TSO/ISPF Client Gateway is used.

Notes:

- The TSO/ISPF Client Gateway will add /WORKAREA to the path specified in _CMDSERV_WORK_HOME. Do not add it yourself.
- If you did not use the SFEKSAMP(FEKSETUP) sample job to build the customizable environment, then you should verify that the WORKAREA directory exists in the path specified in _CMDSERV_WORK_HOME. The directory permission bits must be 777.

STEPLIB

STEPLIB is described previously in the required definitions section.

RSE_CMDSERV_OPTS

Additional TSO/ISPF Client Gateway specific Java options. The default is "". See "Defining extra Java startup parameters with _RSE_CMDSERV_OPTS" on page 41 for more information on this definition. This directive is only required when ISPF's TSO/ISPF Client Gateway is used.

The following definitions are required if SCLM Developer Toolkit is used.

SCLMDT_CONF_HOME

SCLM Developer Toolkit base configuration directory. The default is /var/rdz/scldmt. Change to match the location of the CONFIG directory used by SCLMDT to store SCLM project information. This directive is only required when SCLMDT is used.

Note: SCLMDT will add /CONFIG and /CONFIG/PROJECT to the path specified in SCLMDT_CNF_HOME. Do not add it yourself.

STEPLIB

STEPLIB is described previously in the required definitions section.

_SCLMDT_TRANTABLE

Name of the long/short name translation VSAM. The default is FEK.#CUST.LSTRANS.FILE. Uncomment and change to match the name used in the SCLM sample job ISP.SISPSAMP(FLM02LST). This directive is only required if the long/short name translation in SCLM Developer Toolkit is used.

ANT_HOME

Home directory for your Ant installation. The default is /usr/lpp/apache/Ant/apache-ant-1.7.1. Change to match your Ant installation. This directive is only required when the JAVA/J2EE build support is used with SCLM Developer Toolkit.

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 "Defining the PORTRANGE available for RSE server" on page 36 for more information on this definition. This is an optional directive.

_BPXK_SETIBMOPT_TRANSPORT

Specifies the name of the TCP/IP stack to be used. The default is TCPIP. Uncomment and change to the requested TCP/IP stack name, as defined in the TCPIPJOBNAME statement in the related TCPIP.DATA. This is an optional directive.

Note:

- Coding a SYSTCPD DD statement in the server JCL does not set the requested stack affinity.
- When this directive is not active, RSE binds to every available stack on the system (BIND INADDRANY).

_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. This is an optional directive.

_FEKFSCMD_PARTNER_LU_

Force RSE server to use this APPC partner LU. The default is the base LU specified during APPC configuration. This is an optional directive.

GSK_CRL_SECURITY_LEVEL

Specifies the level of security SSL applications will use when contacting LDAP servers to check CRLs for revoked certificates during certificate validation. The default is MEDIUM. Uncomment and change to enforce the usage of the specified value. This is an optional directive. The following values are valid:

- LOW - Certificate validation will not fail if the LDAP server cannot be contacted.
- MEDIUM - Certificate validation requires the LDAP server to be contactable, but does not require a CRL to be defined. This is the default
- HIGH - Certificate validation requires the LDAP server to be contactable and a CRL to be defined.

Note: This directive requires z/OS 1.9 or higher.

GSK_LDAP_SERVER

Specifies one or more blank-separated LDAP server host names.

Uncomment and change to enforce the usage of the specified LDAP servers to obtain their CRL. This is an optional directive.

The host name can either be a TCP/IP address or an URL. Each host name can contain an optional port number separated from the host name by a colon (:).

GSK_LDAP_PORT

Specifies the LDAP server port. The default is 389. Uncomment and change to enforce the usage of the specified value. This is an optional directive.

GSK_LDAP_USER

Specifies the distinguished name to use when connecting to the LDAP server. Uncomment and change to enforce the usage of the specified value. This is an optional directive.

GSK_LDAP_PASSWORD

Specifies the password to use when connecting to the LDAP server.

Uncomment and change to enforce the usage of the specified value. This is an optional directive.

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

_CEE_RUNOPTS

Language Environment (LE) runtime options. The default is "ALL31(ON) HEAP(32M,32K,ANYWHERE,KEEP,,) TRAP(ON)". 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.

JAVA_PROPAGATE

Propagates the security and workload context during thread creation (Java version 1.4 and older only). The default is NO. Do not modify.

RSE_LIB

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

PATH

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

LIBPATH

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

CLASSPATH

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

_RSE_CMDSERV_OPTS

Additional TSO Commands 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 org.eclipse.dstore.core.server.Server. Do not modify.

_RSE_DAEMON_CLASS

Java class for the RSE daemon. The default is com.ibm.ertools.zos.server.RseDaemon. Do not modify.

_RSE_POOL_SERVER_CLASS

Java class for the RSE thread pool. The default is com.ibm.ertools.zos.server.ThreadPoolProcess. Do not modify.

_RSE_LOCKD_CLASS

Java class for the RSE lock daemon. The default is com.ibm.ftt.rse.mvs.server.miners.MVSLockDaemon. 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.

SCLMDT_BASE_HOME

Home directory for SCLM Developer Toolkit code. The default is \$RSE_HOME. Do not modify.

SCLMDT_WORK_HOME

SCLM Developer Toolkit base work directory. The default is \$_CMDSERV_WORK_HOME. Do not modify.

CGI_DTWORK

SCLM Developer Toolkit support for older clients. The default is \$_SCLMDT_WORK_HOME. Do not modify.

Defining the PORTRANGE available for RSE server

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 port.

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

1. The client connects to host port 4035, RSE daemon.

2. The RSE daemon creates an RSE server thread.
3. The RSE server 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 RSE daemon returns the port number to the client.
5. The client connects to the host port.

Note:

- The process is similar for the (optional) alternative connection method using REXEC/SSH.
- Refer to Chapter 11, “Understanding Developer for System z,” on page 177 for more information.

To specify the port range, for the client to communicate with z/OS, uncomment and customize 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.

The format of `PORTRANGE` is: `_RSE_PORTRANGE=min-max` (max is non-inclusive; for example `_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, then the specified port number is used. If the port is not available connect will fail. This setup is not recommended.

Note: The host can deny this type of connection request by specifying the `deny.nonzero.port=true` directive in `rsed.envvars`. Refer to “Defining extra Java startup parameters with `_RSE_JAVAOPTS`” for more information on this directive.

2. If the port number in the subsystem properties is 0, and if `_RSE_PORTRANGE` is specified in `rsed.envvars`, then the port range specified by `_RSE_PORTRANGE` is used. If no port in the range is available, connect will fail.
3. If the port number in the subsystem properties is 0, and `_RSE_PORTRANGE` is not specified in `rsed.envvars`, then 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.

Defining extra Java startup parameters with `_RSE_JAVAOPTS`

With the different `_RSE_*OPTS` directives, `rsed.envvars` provides the possibility to give extra parameters to Java when it starts the RSE processes. The sample options included in `rsed.envvars` can be activated by uncommenting them.

`_RSE_JAVAOPTS` defines standard and RSE-specific Java options.

```
_RSE_JAVAOPTS=""
```

Variable initialization. Do not modify.

`_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Xms1m -Xmx256m"`

Set initial (Xms) and maximum (Xmx) heap size. The defaults are 1M and 256M respectively. Change to enforce the desired heap size values. If this directive is commented out, the Java default values will be used, which are 4M and 512M respectively (1M and 64M for Java 5.0).

Note: Refer to “Key resource definitions” on page 212 to determine the optimal values for this directive.

`_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Ddaemon.log=/var/rdz/logs"`

Directory holding the RSE daemon and server logging and RSE audit data. The default is /var/rdz/logs. Change to enforce the desired location. If this directive is commented out, the home directory of the user ID assigned to RSE daemon will be used. The home directory is defined in the OMVS security segment of the user ID.

Note: If this directive (or its counterpart, the home directory) does not specify an absolute path (the path does not start with a forward slash (/)), then the actual log location is relative to the configuration directory (by default /etc/rdz).

`_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Duser.log=/var/rdz/logs"`

Directory leading to the user-specific logs. The default is /var/rdz/logs. Change to enforce the desired location. If this directive is commented out, the home directory of the client user ID will be used. The home directory is defined in the OMVS security segment of the user ID.

Note:

- If this directive (or its counterpart, the home directory) does not specify an absolute path (the path does not start with a forward slash (/)), then the actual log location is relative to the configuration directory (by default /etc/rdz).
- The complete path to the user logs is userlog/dstorelog/\$LOGNAME/, where userlog is the value of the user.log directive, dstorelog is the value of the DSTORE_LOG_DIRECTORY directive and \$LOGNAME is the client's user ID in uppercase.
- Ensure that the permission bits for userlog/dstorelog are set so that each client can create \$LOGNAME.

`_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DDSTORE_LOG_DIRECTORY=`

" This directory is appended to the path specified in the user.log directive. Together they create the path leading to the user-specific logs. The default is a null-string. Change to enforce the usage of the specified directory. If this directive is commented out, .eclipse/RSE/ will be used.

Note:

- The complete path to the user logs is userlog/dstorelog/\$LOGNAME/, where userlog is the value of the user.log directive, dstorelog is the value of the DSTORE_LOG_DIRECTORY directive, and \$LOGNAME is the client's user ID in uppercase.
- The directory specified here is relative to the directory specified in user.log, and thus may not start with a forward slash (/).
- Ensure that the permission bits for userlog/dstorelog are set so that each client can create \$LOGNAME.

The following directives are commented out by default.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -Dmaximum.clients=60"

Maximum amount of clients serviced by one thread pool. The default is 60. Uncomment and customize to limit the number of clients per thread pool. Note that other limits may prevent RSE from reaching this limit.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -Dmaximum.threads=1000"

Maximum amount of active threads in one thread pool to allow new clients. The default is 1000. Uncomment and customize to limit the number of clients per thread pool based on the number of threads in use. Note that each client connection uses multiple threads (16 or more) and that other limits may prevent RSE from reaching this limit.

Note: This value must be lower than the setting for MAXTHREADS and MAXTHREADTASKS in SYS1.PARMLIB(BPXPRMxx).

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -Dminimum.threadpool.process=1"

The minimum number of active thread pools. The default is 1. Uncomment and customize to start at least the listed number of thread pool processes. Thread pool processes are used for load balancing the RSE server threads. More new processes are started when they are needed. Starting the new processes up front helps prevent connection delays but uses more resources during idle times.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -Dmaximum.threadpool.process=100"

The maximum number of active thread pools. The default is 100. Uncomment and customize to limit the number of thread pool processes. Thread pool processes are used for load balancing the RSE server threads, so limiting them will limit the amount of active client connections.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -Dipv6=true"

TCP/IP version. The default is false, which means that an IPv4 interface will be used. Uncomment and specify true to use an IPv6 interface.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -Dkeep.last.log=true"

Keep a copy of the host log files belonging to the previous session. The default is false. Uncomment and specify true to rename the previous log files to *.last during server startup and client connect.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -Denable.standard.log=true"

Write the stdout and stderr streams of the thread pools to a log file. The default is false. Uncomment and specify true to save the stdout and stderr streams. The resulting log files are located in the directory referenced by the daemon.log directive.

Note:

- The **MODIFY RSESTANDARDLOG** operator command can be used to dynamically stop or start the update of the stream log files.
- There are no user-specific stdout.log and stderr.log log files when the enable.standard.log directive is active. The user-specific data is now written to the matching RSE thread pool stream.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -Denable.port.of.entry=true"

Port Of Entry (POE) check option. The default is false. Uncomment and specify true to enforce POE checking for client connections. During POE checking, the IP address of the client is mapped into a network access security zone by your security software. The client user ID must have permission to use the profile that defines the security zone.

Note:

- POE checking must also be enabled in your security product.
- Enabling POE checking will enable it for other z/OS UNIX services also, such as INETD.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -Denable.certificate.mapping=false"

Use your security software to authenticate a logon with a X.509 certificate. The default is true. Uncomment and specify false to have RSE daemon do the authentication without relying on the X.509 support of your security software.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -Denable.automount=true"

Support home directories created by z/OS UNIX automount. The default is false. Uncomment and specify true to ensure that z/OS UNIX automount uses the client user ID as owner of the directory.

Note: z/OS UNIX automount uses the user ID of the process that invoked the service when creating a file system. If this option is disabled, this process is the RSE thread pool server (user ID STCRSE). If this option is enabled, a new, temporary process is created using the client user ID before invoking the service.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -Denable.audit.log=true"

Audit option. The default is false. Uncomment and specify true to enforce audit logging of actions done by clients. Audit logs are written to the RSE daemon log location. See the daemon.log option of the _RSE_JAVAOPTS variable to know where this is.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -Daudit.cycle=30"

Number of days stored in 1 audit log file. The default is 30. Uncomment and customize to control how much audit data is written to 1 audit log file. The maximum value is 365.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -Daudit.retention.period=0"

Number of days audit logs are kept. The default is 0 (no limit). Uncomment and customize to delete audit logs after a given number of days. The maximum value is 365.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -Ddeny.nonzero.port=true"

Disallow the client to choose the communication port number. The default is false. Uncomment and specify true to refuse connections where the client specifies which host port must be used by RSE server for the connection. Refer to "Defining the PORTRANGE available for RSE server" on page 36 for more information.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -Dsingle.logon=false"

Disallow a user ID to log on multiple times. The default is true. Uncomment and specify false to allow a user ID to log on multiple times to a single RSE daemon.

Note: A second logon attempt will cause the first one to be cancelled by the host if this directive is not active or set to false. This cancel will be accompanied by console message FEK210I.

RSE_JAVAOPTS="\$_RSE_JAVAOPTS -Dprocess.cleanup.interval=0"

Automatically remove RSE thread pools that are in an unrecoverable error state. By default, erroneous RSE thread pools are not automatically

removed. Uncomment and customize to automatically remove erroneous RSE thread pool servers at every interval (interval unit is seconds). Specifying 0 disables the function.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -DAPPLID=FEKAPPL"

RSE server application ID. The default is FEKAPPL. Uncomment and customize this option to enforce the use of the desired application ID.

Note:

- The application ID must be defined to your security software. Failure to do so will prevent the client from logging on.
- Refer to “Using PassTickets” on page 152 for the security implications when changing this value.
- The application ID must match the application ID used by JES Job Monitor. Refer to “FEJJCNFG, JES Job Monitor configuration file” on page 24 to learn how to define the application ID for JES Job Monitor.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -DDENY_PASSWORD_SAVE=true"

Password save option. The default is false. Uncomment and specify true 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 higher.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -DHIDE_ZOS_UNIX=true"

Hide z/OS UNIX option. The default is false. Uncomment and specify true to prevent users from seeing z/OS UNIX elements (directory structure and command line) on the client. This option only works with clients version 7.6 and higher.

**#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS
-DDSTORE_IDLE_SHUTDOWN_TIMEOUT=3600000"**

Disconnect idle clients. By default, idle clients are not disconnected. Uncomment and customize to disconnect clients who are idle for the listed amount of milliseconds (3600000 equals 1 hour).

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -DDSTORE_TRACING_ON=true"

Start dstore tracing. Use only when directed by the IBM support center. Note that the resulting .dstoreTrace log file is created in Unicode (ASCII), not EBCDIC.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -DDSTORE_MEMLOGGING_ON=true"

Start dstore memory tracing. Use only when directed by the IBM support center. Note that the resulting .dstoreMemLogging log file is created in Unicode (ASCII), not EBCDIC.

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

Use an APPC transaction for the TSO Commands service. By default, ISPF's TSO/ISPF Client Gateway is used. Uncomment to use an APPC transaction instead. Do not change the assigned value.

Defining extra Java startup parameters with _RSE_CMDSERV_OPTS

With the different _RSE_*OPTS directives, rsed.envvars provides the possibility to give extra parameters to Java when it starts the RSE processes. The sample options included in rsed.envvars can be activated by uncommenting them.

The `_RSE_CMDSERV_OPTS` directives are RSE-specific Java options and are only in effect when ISPF's TSO/ISPF Client Gateway is used by the Developer for System z. (This is the default.)

`_RSE_CMDSERV_OPTS=""`

Variable initialization. Do not modify.

`_RSE_CMDSERV_OPTS="$_RSE_CMDSERV_OPTS &ISPROF=
&SYSUID..ISPROF="`

Use an existing ISPF profile for the ISPF initialization. Uncomment and change the data set name to use the specified ISPF profile.

The following variables can be used in the data set name:

- `&SYSUID`, to substitute the developer's user ID
- `&SYSPREF`, to substitute the developer's TSO prefix

ISPF.conf, ISPF's TSO/ISPF Client Gateway configuration file

ISPF's TSO/ISPF Client Gateway uses the definitions in `ISPF.conf` to create a valid environment to execute batch TSO and ISPF commands. Developer for System z uses this environment to run some MVS based services. These services include the TSO Commands service, SCLM Developer Toolkit service and an alternate CARMA startup method.

`ISPF.conf` is located in `/etc/rdz/`, unless you specified a different location when you customized and submitted job `FEK.SFEKSAMP(FEKSETUP)`. See "Customization setup" on page 13 for more details. You can edit the file with the TSO `OEDIT` command.

Comment lines start with an asterisk (*) when using a US code page. Data lines can only have a directive and its assigned value. Comments are not allowed on the same line. Line continuations are not supported. When concatenating data set names, add them on the same line and separate the names with a comma (,).

In addition to providing the correct names for the ISPF data sets, you must also add the TSO Commands service data set name, `FEK.SFEKPROC`, to the `SYSPROC` or `SYSEXEC` statement, as shown in the following example.

```
* REQUIRED:
sysproc=ISP.SISPCLIB,FEK.SFEKPROC
ispm1ib=ISP.SISPMENU
ispt1ib=ISP.SISPTENU
ispp1ib=ISP.SISPPENU
isps1ib=ISP.SISPSLIB
ispl1ib=ISP.SISLOAD

* OPTIONAL:
*allocjob = FEK.#CUST.CNTL(CRAISPRX)
*ISPF_timeout = 900
```

Figure 9. ISPF.conf - ISPF configuration file

Note:

- You can add your own DD-like statements and data set concatenations to customize the TSO environment, thus mimicking a TSO logon procedure. See Chapter 16, "Customizing the TSO environment," on page 243 for more details.
- The TSO/ISPF Client Gateway may not function properly if you use a (third party) product that intercepts ISPF commands, such as **ISPSTART**.

Check the documentation for that product on how it can be disabled for Developer for System z. If the product requires the allocation of a specific DD statement to DUMMY, you can simulate this in ISPF.conf by allocating that DD statement to nullfile.

For example:

```
ISPTRACE=nullfile
```

- When using the allocjob directive, be careful not to undo the DD definitions done earlier in ISPF.conf.
- System abend 522 for module ISPZTS0 is to be expected if the JWT parameter in the SMFPRMxx parmlib member is set lower than the ISPF_timeout value in ISPF.conf. This does not impact Developer for System z operations, as the TSO/ISPF Client Gateway is restarted automatically when needed.
- Changes are active for all new invocations. No server restart is needed.

Optional components

The customization steps above are for a basic Developer for System z setup. Refer to the chapters about the optional components for their customization requirements:

- Chapter 3, “(Optional) Common Access Repository Manager (CARMA),” on page 45
- Chapter 4, “(Optional) Application Deployment Manager,” on page 65
- Chapter 5, “(Optional) SCLM Developer Toolkit,” on page 73
- “(Optional) DB2 stored procedure” on page 81
- “(Optional) CICS bidirectional language support” on page 84
- “(Optional) RSE SSL encryption” on page 85
- “(Optional) RSE tracing” on page 88
- “(Optional) Host based property groups” on page 89
- “(Optional) Host based projects” on page 90
- “(Optional) File Manager integration” on page 91
- “(Optional) Uneditable characters” on page 92
- “(Optional) Using REXEC (or SSH)” on page 93
- “(Optional) APPC transaction for the TSO Commands service” on page 95
- “(Optional) WORKAREA cleanup” on page 98

Installation verification

The description of the various installation verification programs (IVPs) is located in Chapter 7, “Installation verification,” on page 99, because some of the IVPs are for the optional components.

Chapter 3. (Optional) Common Access Repository Manager (CARMA)

Common Access Repository Manager (CARMA) is a productivity aid for developers who are creating Repository Access Managers (RAMs). A RAM is an Application Programming Interface (API) for z/OS based Software Configuration Managers (SCMs).

In turn, user-written applications can start a CARMA server which loads the RAMs and provides a standard interface to access the SCM.

Developer for System z supports multiple methods to start a CARMA server, each with their own benefits and drawbacks.

- The “batch submit” method starts the CARMA server by submitting a job. This is the default method used in the provided sample configuration files. The benefit of this method is that the CARMA logs are easily accessible in the job output. It also allows the use of custom server JCL for each developer, which is maintained by the developer himself. However, this method uses one JES initiator per developer starting a CARMA server.
- The “CRASTART” method starts the CARMA server as a subtask within RSE. It provides a very flexible setup by using a separate configuration file that defines data set allocations and program invocations needed to start a CARMA server. This method provides the best performance and uses the fewest resources, but requires that module CRASTART is located in LPA.
- The “TSO/ISPF Client Gateway” method uses ISPF’s TSO/ISPF Client Gateway to create a TSO or ISPF environment, in which the CARMA server is started. It allows for flexible data set allocations using the possibilities of ISPF.conf. However, this method is not suited to access SCMs that interfere with normal TSO or ISPF operations.

Requirements and checklist

You will need the assistance of a security administrator and a TCP/IP administrator to complete this customization task, which requires the following resources or special customization tasks:

- TCP/IP port range for internal communication
- Security rule to allow developers update to CARMA VSAM files
- (Optional) Security rule to allow users to submit CRA* jobs
- (Optional) LPA update

In order to start using CARMA at your site, you must perform the following tasks. Unless otherwise indicated, all tasks are mandatory.

1. Create required CARMA components. For details, see “CARMA components” on page 46.
2. Initial customization of RSE configuration files to interface with CARMA. The complete customization is dependent on the method chosen to start CARMA. For details, see “RSE interface to CARMA” on page 47.
3. Choose a method to start CARMA and do the required customization of the related configuration files. For details see:
 - “CARMA server startup using batch submit” on page 49

- “(Optional) Alternative CARMA server startup using CRASTART” on page 50
 - “(Optional) Alternative CARMA server startup using TSO/ISPF Client Gateway” on page 53
4. Optionally activate sample Repository Access Managers (RAMs). For details see “(Optional) Activating the sample Repository Access Managers (RAMs)” on page 55.
 5. Optionally activate CA Endeavor® RAM. For details see “(Optional) Activating the CA Endeavor® SCM RAM” on page 56.
 6. Optionally create CRAXJCL as replacement for IRXJCL. For details, see “(Optional) IRXJCL versus CRAXJCL” on page 63.

Note: The sample members referenced in this chapter are located in FEK.#CUST.* and /etc/rdz, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details.

CARMA components

The following CARMA components must be customized, regardless of the chosen startup method. The sample members referenced below are located in FEK.#CUST.JCL, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details.

1. Customize and submit the FEK.#CUST.JCL(CRA\$VDEF) JCL. Refer to the documentation within CRA\$VDEF for customization instructions. CRA\$VDEF creates and primes the CARMA configuration VSAM data set, CRADEF.
2. Customize and submit the FEK.#CUST.JCL(CRA\$VMSG) JCL. Refer to the documentation within CRA\$VMSG for customization instructions. CRA\$VMSG creates and primes the CARMA message VSAM data set, CRAMSG.
3. Customize and submit the FEK.#CUST.JCL(CRA\$VSTR) JCL. Refer to the documentation within CRA\$VSTR for customization instructions. CRA\$VSTR creates and primes the CARMA custom information VSAM data set, CRASTRS.

Note:

- The CARMA VSAMs created with these jobs define the sample RAMs. Refer to “(Optional) Activating the CA Endeavor® SCM RAM” on page 56 to define the CA Endeavor® RAM.
- Refer to sample job FEK.#CUST.JCL(CRA#UADD) if you need to merge the definitions for a (custom) RAM into an existing VSAM configuration. This job must be customized and submitted for each CARMA VSAM that changes. Refer to the *Rational Developer for System z Common Access Repository Manager Developer's Guide* (SC23-7660) for more information on the record structure used by the different CARMA VSAMs.
- Use sample job FEK.#CUST.JCL(CRA#UQRY) to extract the active definitions from a VSAM to a sequential data set.

CARMA VSAM migration notes

Developer for System z version 7.6.1 supports a new data structure layout for the CARMA custom information VSAM data set, CRASTRS, to remove message length limitations.

Prior to Developer for System z version 7.6.1, strings defined in the CARMA custom information VSAM data set are limited to predefined lengths. This limitation forces RAM developers to shorten descriptive strings, or to use client-side plug-ins to display full-length strings.

Developer for System z version 7.6.1 supports a new, variable-length, data structure layout for the CARMA custom information VSAM data set, CRASTRS, where strings are separated by a delimiter character instead of being fixed length.

Customize and submit the FEK.SFEKSAMP(CRA#VS2) JCL to convert your existing, fixed-length, CARMA custom information VSAM data set, CRASTRS, to the new variable-length format.

Note:

- Beginning with version 7.6.1, the sample CARMA custom information VSAM data set is shipped in variable-length format.
- Beginning with version 7.6.1, the CARMA load module, CRASERV, supports both the fixed-length format and the variable-length format for the CARMA custom information VSAM data set.
- Older versions of the CARMA load module do not support the variable-length format and will produce garbled strings when used with a variable-length CARMA custom information VSAM data set.

RSE interface to CARMA

The CARMA server provides a standard API for other host-based products to access one or more Software Configuration Managers (SCMs). However, it does not provide methods for direct communication with a client PC. For this, it relies on other products, such as the RSE server. The RSE server uses the settings in CRASRV.properties to start and connect to a CARMA server.

CRASRV.properties is located in /etc/rdz/, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details. You can edit the file with the TSO **OEDIT** command.

Note: The RSED started task must be restarted to pick up any changes you make.

```
# CRASRV.properties - CARMA configuration options
#
port.start=5227
port.range=100
startup.script.name=/usr/lpp/rdz/bin/carma.startup.rex
clist.dsname='FEK.#CUST.CNTL(CRASUBMT)'
crastart.stub=/usr/lpp/rdz/bin/CRASTART
crastart.configuration.file=/etc/rdz/crastart.conf
crastart.syslog=Partial
crastart.timeout=420
#crastart.step1lib=FEK.SFEKLPA
#crastart.tasklib=TASKLIB
```

Figure 10. CRASRV.properties – CARMA configuration file

port.start

First port used for communication between CARMA and the RSE server. 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 140 for more information.

port.range

Range of ports, starting at `port.start`, which will be used for CARMA 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 CARMA startup script. The default is `/usr/lpp/rdz/bin/carma.startup.rex`. This REXX exec will trigger the startup of a CARMA server.

clist.dsname

Defines the startup method for the CARMA server.

- `*CRASTART` indicates that the CARMA server should be started as a subtask within RSE using `CRASTART`. Refer to “(Optional) Alternative CARMA server startup using `CRASTART`” on page 50 for more details. If you specify `*CRASTART`, you must also specify the `crastart.*` directives.
- `*ISPF` indicates that the CARMA server should be started using ISPF’s TSO/ISPF Client Gateway. Refer to “(Optional) Alternative CARMA server startup using TSO/ISPF Client Gateway” on page 53 for more details.
- Any other value defines the location of the `CRASUBMT CLIST`, using TSO-like naming conventions. With quotes (') the data set name is an absolute reference, without quotes (') the data set name is prefixed with the client’s user ID, not the TSO prefix. The latter requires that all CARMA users must maintain their own `CRASUBMT CLIST`.

The default is `'FEK.#CUST.CNTL(CRASUBMT)'`. This CLIST will start a CARMA server when opening a connection using the batch submit method.

crastart.stub

z/OS UNIX stub for calling `CRASTART`. The default is `/usr/lpp/rdz/bin/CRASTART`. This stub makes the MVS based `CRASTART` load module available to z/OS UNIX processes. This directive is only used if the `clist.dsname` directive has `*CRASTART` as value.

crastart.configuration.file

Specifies the name of the `CRASTART` configuration file. The default is `/etc/rdz/crastart.conf`. This file specifies the data set allocations and program invocations needed to start a CARMA server. This directive is only used if the `clist.dsname` directive has `*CRASTART` as value.

crastart.syslog

Specifies how much information is written to the system log while `CRASTART` starts a CARMA server. The default is `Partial`. Valid values are:

A (All)	All tracing information is printed to SYSLOG
P (Partial)	Only connect, disconnect, and error information is printed to SYSLOG
anything else	Only error conditions are printed to SYSLOG

This directive is only used if the `clist.dsname` directive has `*CRASTART` as value.

crastart.timeout

The length of time, in seconds, before a CARMA server ends due to lack of activity. The default is 420 (7 minutes). This directive is only used if the `clist.dsname` directive has `*CRASTART` as value.

crastart.steplib

The location of the CRASTART module when accessed through the STEPLIB directive in `rsed.envvars`. The default is `FEK.SFEKLPA`.

Uncomment and customize this directive if the CRASTART module cannot be part of LPA or LINKLIST. Note that program control and APF issues may arise if the CRASTART module is not in LPA. This directive is only used if the `clist.dsname` directive has `*CRASTART` as value.

crastart.tasklib

Alternate name for the TASKLIB DD name in `crastart.conf`. The default is TASKLIB. Uncomment and customize this directive if DD name TASKLIB has a special meaning for your SCM or RAM and cannot be used as STEPLIB replacement. This directive is only used if the `clist.dsname` directive has `*CRASTART` as value.

CARMA server startup using batch submit

The information in this section describes how to configure the default method for Developer for System z to start a CARMA server. This customization step can be bypassed if you use another startup method.

Developer for System z uses by default the batch submit CARMA server startup method that does not require the CRASTART module to be in LPA and does not depend on the TSO/ISPF Client Gateway. The method submits the CARMA server as a long-running batch job in your JES.

Adjust CRASRV.properties

RSE server uses the settings in `/etc/rdz/CRASRV.properties` to start and connect to a CARMA server, as documented in “RSE interface to CARMA” on page 47. You can edit the file with the TSO **OEDIT** command. Note that RSE must be restarted for the changes to take effect.

Change the value of the `clist.dsname` directive to the data set and member name of the CRASUBMT CARMA server startup CLIST, as shown in the following example. Refer to “RSE interface to CARMA” on page 47 for more information on the different directives.

```
port.start=5227
port.range=100
startup.script.name=/usr/lpp/rdz/bin/carma.startup.rex
clist.dsname='FEK.#CUST.CNTL(CRASUBMT)'
```

Figure 11. CRASRV.properties - CARMA startup using batch submit

Adjust CRASUBMT

Customize the CRASUBMT CLIST, as shown in the following code sample. Refer to the documentation within CRASUBMT for customization instructions. The CRASUBMT CLIST submits a CARMA server.

CRASUBMT is located in FEK.#CUST.CNTL, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details.

```
PROC 1 PORT TIMEOUT(420)
SUBMIT * END($$)
//CRA&PORT JOB CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1)
//RUN      EXEC PGM=IKJEFT01,DYNAMNBR=25,REGION=1024K,TIME=NOLIMIT
//STEPLIB DD DISP=SHR,DSN=FEK.SFEKLOAD
//*        DD DISP=SHR,DSN=FEK.#CUST.LOAD
//CRADEF   DD DISP=SHR,DSN=FEK.#CUST.CRADEF
//CRAMSG   DD DISP=SHR,DSN=FEK.#CUST.CRAMSG
//CRASTRS  DD DISP=SHR,DSN=FEK.#CUST.CRASTRS
//*CRARAM1 DD DISP=SHR,DSN=FEK.#CUST.CRARAM1
//*
//ISPPROF DD DISP=(NEW,DELETE,DELETE),
//          SPACE=(TRK,(1,1,5)),LRECL=80,RECFM=FB,UNIT=SYSALLDA
//ISPMLIB DD DISP=SHR,DSN=ISP.SISPMENU
//ISPPLIB DD DISP=SHR,DSN=ISP.SISPPENU
//ISPSLIB DD DISP=SHR,DSN=ISP.SISPSENU
//ISPTLIB DD DISP=SHR,DSN=ISP.SISPTENU
//ISPEXEC DD DISP=SHR,DSN=ISP.SISPEXEC
//SYSPROC DD DISP=SHR,DSN=ISP.SISPCLIB
//*
//CARMALOG DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
ISPSTART PGM(CRASERV) PARM(&PORT &TIMEOUT)
//*
$$
EXIT CODE(0)
```

Figure 12. CRASUBMT - CARMA startup using batch submit

Note:

- You can add your own DD statements and data set concatenations to customize the CARMA TSO environment, thus mimicking a TSO logon procedure.
- You can optionally change CARMA's timeout value by modifying the PROC 1 PORT TIMEOUT(420) line in FEK.#CUST.CNTL(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).
- Details of the CARMA startup process are shown in rsecomm.log. Refer to “(Optional) RSE tracing” on page 88 for more information on setting the detail level of rsecomm.log.
- Changes are in effect for all CARMA servers started after the update.

(Optional) Alternative CARMA server startup using CRASTART

The information in this section describes how to configure an alternative method for Developer for System z to start a CARMA server. This customization step can be bypassed if you use another startup method.

Developer for System z supports an alternative CARMA server startup method that does not depend on the TSO/ISPF Client Gateway and that does not submit a server job using a JES initiator. The method uses CRASTART to start the CARMA server as a subtask within RSE and is similar to the TSO/ISPF Client Gateway service.

Note: Details of the CARMA startup process are shown in `rsecomm.log`. Refer to “(Optional) RSE tracing” on page 88 for more information on setting the detail level of `rsecomm.log`.

Adjust CRASRV.properties

RSE server uses the settings in `/etc/rdz/CRASRV.properties` to start and connect to a CARMA server, as documented in “RSE interface to CARMA” on page 47. You can edit the file with the TSO **OEDIT** command. Note that RSE must be restarted for the changes to take effect.

Change the value of the `clist.dsname` directive to `*CRASTART` and provide the correct values for the `crastart.*` directives, as shown in the following example. Refer to “RSE interface to CARMA” on page 47 for more information on the different directives.

```
port.start=5227
port.range=100
startup.script.name=/usr/lpp/rdz/bin/carma.startup.rex
clist.dsname=*CRASTART
crastart.stub=/usr/lpp/rdz/bin/CRASTART
crastart.configuration.file=/etc/rdz/crastart.conf
crastart.syslog=Partial
crastart.timeout=420
#crastart.steplib=FEK.SFEKLPA
#crastart.tasklib=TASKLIB
```

*Figure 13. CRASRV.properties - *CRASTART alternative CARMA startup*

Note: System abend 522 for module CRASERV will occur if the `JWT` parameter in the `SMFPRMxx parmlib` member is set to a value lower than the time out value in `CRASRV.properties`. This does not impact CARMA operations, as the server is restarted automatically if needed.

Adjust crastart.conf

Keep a printout of the customized `CRASUBMT` (see “CARMA server startup using batch submit” on page 49) handy for easy reference during this customization step. The printout will be valuable even if you have not customized the member.

`CRASTART` uses the definitions in `crastart.conf` to create a valid environment to execute batch TSO and ISPF commands. Developer for System z uses this environment to run the CARMA server called `CRASERV`.

`crastart.conf` is located in `/etc/rdz/`, unless you specified a different location when you customized and submitted job `FEK.SFEKSAMP (FEKSETUP)`. See “Customization setup” on page 13 for more details. You can edit the file with the TSO **OEDIT** command.

Note: Changes are in effect for all CARMA servers started after the update.

The following customization steps are needed to adjust the configuration file shown in the code sample below.

- Add the data sets allocated to the `STEPLIB` concatenation of the `CRASUBMT` procedure to the `TASKLIB` statement in `crastart.conf`.
- Create entries for the mandatory CARMA VSAM DD's, `CRADef`, `CRAMSG`, and `CRSTRS`. Use the data set names provided in the (customized) `CRASUBMT` procedure.

- Add any custom DD statement and the related data set concatenations, available in the (customized) CRASUBMT procedure. For example, add the CRARAM1 DD statement and data set name if you use the sample PDS RAM. Note that you can use data set names (allocated with DISP=SHR), SYSOUT and DUMMY constructs.
- Optionally add any BPXWDYN command using the -COMMAND statement. There can be multiple -COMMAND statements. BPXWDYN allows you to do more complex allocations, including creating temporary data sets, dispositions other than SHR, allocations to other subsystems, and so on. Refer to *Using REXX and z/OS UNIX System Services* (SA22-7806) for more information on BPXWDYN.
- Select the desired program invocation method using the PROGRAM statement. The advised method is "PROGRAM=IKJEFT01 CRASERV &CRAPRM1. &CRAPRM2.", as it gives you a TSO environment that can handle a mixture of APF and non-APF data sets. See the sample crastart.conf for other methods.

Note: crastart.conf definitions cannot be split across multiple lines.

* crastart.conf - CARMA allocation options

```
TASKLIB = FEK.SFEKLOAD
CRADEF = FEK.#CUST.CRADEF
CRAMSG = FEK.#CUST.CRAMSG
CRASTRS = FEK.#CUST.CRASTRS
*CRARAM1 = FEK.#CUST.CRARAM1
*
CARMALOG = SYSOUT(H)
SYSTSPRT = SYSOUT(H)
SYSTSIN = DUMMY
-COMMAND=ALLOC FI(SCRATCH) NEW DELETE DSORG(PS) RECFM(F,B) LRECL(80) UNIT(VIO)
*
PROGRAM=IKJEFT01 CRASERV &CRAPRM1. &CRAPRM2.
```

Figure 14. crastart.conf - *CRASTART alternative CARMA startup

The following variables can be used in the configuration file:

Table 10. crastart.conf variables

&CRAUSER.	Logon user ID of the client.
&CRADATE.	Current [®] date in Dyyyymmdd format (7 character Julian).
&CRATIME.	Current time in Thhmmss format (hour minute second).
&CRAPRM3. through &CRAPRM9.	Additional variables with user-assigned values. The usage of these variables requires customizing the CARMA startup REXX referenced by startup.script.name in CRASRV.properties. When you use these variables, you should customize a copy of the default startup REXX, /usr/lpp/rdz/bin/carma.startup.rex, and point startup.script.name to this copy. This to avoid losing your work when maintenance updates the default REXX.
system symbol	Any system symbol defined in SYS1.PARMLIB(IEASYMxx)

Table 10. *crastart.conf* variables (continued)

-<DD>	A dash (-) followed by a previously defined DD name acts like a *.ddname backward reference in JCL. The original DD must be allocated using the -COMMAND statement.
-------	---

Note: There is no variable for the TSO prefix, because TSO is not active when the configuration file is interpreted.

(Optional) Alternative CARMA server startup using TSO/ISPF Client Gateway

The information in this section describes how to configure an alternative method for Developer for System z to start a CARMA server. This customization step can be bypassed if you use another startup method.

Developer for System z supports an alternative CARMA server startup method that does not require the CRASTART module to be in LPA and that does not submit a server job using a JES initiator. The method uses ISPF's TSO/ISPF Client Gateway and is similar to the default way of accessing the TSO Commands service.

Note: Details of the CARMA startup process are shown in *rsecomm.log*. Refer to “(Optional) RSE tracing” on page 88 for more information on setting the detail level of *rsecomm.log*.

Adjust CRASRV.properties

RSE server uses the settings in */etc/rdz/CRASRV.properties* to start and connect to a CARMA server, as documented in “RSE interface to CARMA” on page 47. You can edit the file with the TSO **OEDIT** command. Note that RSE must be restarted for the changes to take effect.

Change the value of the *clist.dsname* directive to **ISPF*, as shown in the following example. Refer to “RSE interface to CARMA” on page 47 for more information on the different directives.

```
port.start=5227
port.range=100
startup.script.name=/usr/lpp/rdz/bin/carma.startup.rex
clist.dsname=*ISPF
```

Figure 15. *CRASRV.properties* - **ISPF* alternative CARMA startup

Adjust ISPF.conf

Keep a printout of the customized CRASUBMT (see “CARMA server startup using batch submit” on page 49) handy for easy reference during this customization step. The printout will be valuable even if you have not customized the member.

ISPF's TSO/ISPF Client Gateway uses the definitions in *ISPF.conf* to create a valid environment to execute batch TSO and ISPF commands. Developer for System z uses this environment to run the CARMA server.

ISPF.conf is located in /etc/rdz/, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details. You can edit the file with the TSO OEDIT command.

Note: Changes are in effect for all CARMA servers started after the update.

The following customization steps are needed to adjust the configuration file shown in the code sample below.

- Provide the correct names for the mandatory ISPF data sets (do not allocate ISPPROF however, this one is allocated dynamically).
- Append the Developer for System z proclib, FEK.SFEKPROC, to the SYSPROC or SYSEXEC statement, so that the CRASRVI exec can be found by the system. This exec starts the CARMA server (and thus replaces CRASUBMT’s SYSTSIN DD).
- Append the DD STEPLIB concatenation of the CRASUBMT procedure to the ispllib statement.
- Create entries for the mandatory CARMA VSAM DDs, CRADEF, CRAMSG, and CRSTRS. Use the data set names provided in the (customized) CRASUBMT procedure.
- Add any custom DD statement and the related data set concatenation, available in the (customized) CRASUBMT procedure. For example, add the CRARAM1 DD statement and data set name if you use the sample PDS RAM. Note that you can only use data set names (allocated with DISP=SHR).
- Optionally uncomment and customize the allocexec directive to do additional allocations using an exec.

Note: Do not include the SYSTSIN, SYSTSOUT, or CARMALOG DDs, nor any other DD statement that uses JES constructs such as instream data and SYSOUT=. These entries must be converted to use data sets.

```
sysproc=ISP.SISPLIB,FEK.SFEKPROC
ispllib=FEK.SFEKLOAD
ispmlib=ISP.SISPMENU
isptlib=ISP.SISPTENU
ispplib=ISP.SISPPENU
ispslib=ISP.SISPSLIB
CRADEF =FEK.#CUST.CRADEF
CRAMSG =FEK.#CUST.CRAMSG
CRASTRS=FEK.#CUST.CRASTRS
*CRARAM1=FEK.#CUST.CRARAM1
allocjob=FEK.#CUST.CNTL(CRAISPRX)
```

*Figure 16. ISPF.conf - *ISPF alternative CARMA startup*

DD CARMALOG refers to SYSOUT=* by default, which cannot be mapped in ISPF.conf. You cannot map the DD directly to a data set either, since all Developer for System z users will be using the same ISPF.conf file and thus the same data sets.

However, as described in Chapter 16, “Customizing the TSO environment,” on page 243, section “Advanced – Using an allocation exec” on page 244, you can use an allocation exec to create and allocate a data set based upon the active user ID. See sample member CRAISPRX in data set FEK.#CUST.CNTL as an example that allocates DD CARMALOG to data set name TSOPREFIX'.'USERID'.CRA.'TIMESTAMP'.CARMALOG'.

Note:

- When using the `allocjob` directive, be careful not to undo the DD definitions done earlier in `ISPF.conf`.
- System abend 522 for module CRASERV is to be expected if the `JWT` parameter in the `SMFPRMxx` parmlib member is set lower than the `ISPF_timeout` value in `ISPF.conf`. This does not impact CARMA operations, as the server is restarted automatically if needed.

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

Repository Access Managers (RAMs) are user-written APIs 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.

Refer to *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.

The sample members referenced below are located in `FEK.#CUST.JCL`, unless you specified a different location when you customized and submitted job `FEK.SFEKSAMP(FEKSETUP)`. See “Customization setup” on page 13 for more details.

Activating the PDS RAM

The PDS RAM gives a data set list similar to **MVS Files -> My Data Sets** in the Remote Systems view. The PDS RAM uses RAM ID 0 by default.

Note: The PDS RAM expects that CARMA is started within ISPF (using `ISPSTART`).

1. Customize and submit the `FEK.#CUST.JCL(CRA#VPDS)` JCL. Refer to the documentation within `CRA#VPDS` for customization instructions. `CRA#VPDS` creates and primes the PDS RAM message VSAM data set.
2. Add the `CRARAM1` DD statement to the selected CARMA startup method and provide the data set name of the PDS RAM message VSAM.

Activating the SCLM RAM

The SCLM RAM gives a basic entry into SCLM, ISPF's Software Configuration Manager. The SCLM RAM uses RAM ID 1 by default.

Note: The SCLM RAM expects that CARMA is started within ISPF (using `ISPSTART`).

1. Customize and submit the `FEK.#CUST.JCL(CRA#VSLM)` JCL. Refer to the documentation within `CRA#VSLM` for customization instructions. `CRA#VSLM` creates and primes the SCLM RAM message VSAM data set.
2. Add the `CRARAM2` DD statement to the selected CARMA startup method and provide the data set name of the SCLM RAM message VSAM.
3. Customize the `FEK.#CUST.JCL(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 FEK.#CUST.JCL(CRA#ASLM) once before using CARMA with the SCLM RAM. Failing to do so will result in an allocation error.

Activating the skeleton RAM

The skeleton RAM gives a skeleton framework that can be used to develop your own RAMs. The skeleton RAM uses RAM ID 3 by default.

1. Customize and submit the FEK.#CUST.JCL(CRA#CRAM) JCL. Refer to the documentation within CRA#CRAM for customization instructions. CRA#CRAM compiles the skeleton RAM.
2. Add the load library holding the compiled skeleton RAM module, CRARAMSA, to the STEPLIB DD of the selected CARMA startup method (TASKLIB DD for the CRASTART method).

(Optional) Activating the CA Endeavor® SCM RAM

The IBM® Rational® Developer for System z Interface for CA Endeavor® Software Configuration Manager gives Developer for System z clients direct access to CA Endeavor® SCM. From here on, IBM® Rational® Developer for System z Interface for CA Endeavor® SCM is abbreviated to CA Endeavor® SCM RAM (Repository Access Manager).

In contradiction with the sample RAMs documented in this publication, CA Endeavor® SCM RAM is a production type RAM. You should not activate both types of RAM in the same setup.

Attention: The provided setup jobs for CA Endeavor® SCM RAM replace the active CARMA setup with one that holds only the CA Endeavor® SCM RAM.

Note: The TSO/ISPF Client Gateway startup method can not be used together with the CA Endeavor® SCM RAM.

Requirements and checklist

You need the assistance of a security administrator and a TCP/IP administrator to complete this customization task, which requires the following resources or special customization tasks:

- TCP/IP port range for internal communication
- (Optional) Security rule to allow users to submit CRA* jobs
- (Optional) LPA update

In order to start using the CA Endeavor® SCM RAM at your site, you must perform the following tasks. Unless otherwise indicated, all tasks are mandatory.

1. Allocate and prime VSAM data sets that define the CA Endeavor® SCM RAM to CARMA. For details, see “Define the CA Endeavor® SCM RAM” on page 57.
2. Choose your preferred startup method, batch submit or CRASTART, and do the required customization of the related configuration files. For details see:
 - “CA Endeavor® SCM RAM startup using batch submit” on page 57
 - “CA Endeavor® SCM RAM startup using CRASTART” on page 60
3. Optionally customize the allocation exec used for dynamic allocation of user-specific data sets. For details, see “(Optional) Customize CRANDVRA” on page 61.

4. Optionally customize the CA Endeavor® SCM RAM specific configuration files. For details, see “(Optional) Customize the CA Endeavor® SCM RAM” on page 62.

Define the CA Endeavor® SCM RAM

The following CARMA components must be customized, regardless of the chosen startup method. The sample members referenced below are located in FEK.#CUST.JCL, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details.

1. Customize and submit the FEK.#CUST.JCL(CRA#VCAD) JCL. Refer to the documentation within CRA\$VDEF for customization instructions. CRA#VCAD creates and primes the CARMA configuration VSAM data set, CRADEF.
2. Customize and submit the FEK.#CUST.JCL(CRA\$VMSG) JCL. Refer to the documentation within CRA\$VMSG for customization instructions. CRA\$VMSG creates and primes the CARMA message VSAM data set, CRAMSG.

Note: This is the same job as for the sample RAMs.

3. Customize and submit the FEK.#CUST.JCL(CRA#VCAS) JCL. Refer to the documentation within CRA\$VSTR for customization instructions. CRA#VCAS creates and primes the CARMA custom information VSAM data set, CRASTRS.

Note:

- The CA Endeavor® SCM RAM uses RAM ID 0 by default.
- Refer to sample job FEK.#CUST.JCL(CRA#UADD) if you need to merge the definitions for a (custom) RAM into an existing VSAM configuration. This job must be customized and submitted for each CARMA VSAM that changes. Refer to the *Rational Developer for System z Common Access Repository Manager Developer's Guide* (SC23-7660) for more information on the record structure used by the different CARMA VSAMs.
- Use sample job FEK.#CUST.JCL(CRA#UQRY) to extract the active definitions from a VSAM to a sequential data set.

CA Endeavor® SCM RAM startup using batch submit

Do not execute this step if you use the CRASTART method to start the CARMA server with the CA Endeavor® SCM RAM.

Developer for System z can use the batch submit CARMA server startup method to start the CA Endeavor® SCM RAM. The method submits the CARMA server as a long-running batch job in your JES.

Refer to “CARMA server startup using batch submit” on page 49 for more information on the batch submit startup method.

Adjust CRASRV.properties

RSE server uses the settings in /etc/rdz/CRASRV.properties to start and connect to a CARMA server, as documented in “RSE interface to CARMA” on page 47. You can edit the file with the TSO OEDIT command. Note that RSE must be restarted for the changes to take effect.

Change the value of the clist.dsname directive to the data set and member name of the CRASUBCA CARMA server startup CLIST, as shown in the following example. Refer to “RSE interface to CARMA” on page 47 for more information on the

different directives.

```
port.start=5227
port.range=100
startup.script.name=/usr/lpp/rdz/bin/carma.startup.rex
clist.dsname='FEK.#CUST.CNTL(CRASUBCA)'
```

Figure 17. Figure x1. CRASRV.properties - CA Endeavor® SCM RAM startup using batch submit

Adjust CRASUBCA

Customize the CRASUBCA CLIST, as shown in the following code sample. Refer to the documentation within CRASUBCA for customization instructions. The CRASUBCA CLIST submits a CARMA server for CA Endeavor® SCM.

CRASUBCA is located in FEK.#CUST.CNTL, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details.

```

PROC 1 PORT TIMEOUT(420)
SUBMIT * END($$)
//CRA&PORT JOB CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1)
//RUN      EXEC PGM=IKJEFT01,DYNAMNBR=125,REGION=0M,TIME=NOLIMIT,
//          PARM='%CRANDVRA NDVRC1 PGM(CRASERV) PARM(&PORT &TIMEOUT)'
//STEPLIB DD DISP=SHR,DSN=FEK.SFEKLOAD
//          DD DISP=SHR,DSN=CA.NDVR.AUTHLIB
//          DD DISP=SHR,DSN=CA.NDVRL.AUTHLIB
//CRADEF DD DISP=SHR,DSN=FEK.#CUST.CRADEF
//CRMSG DD DISP=SHR,DSN=FEK.#CUST.CRMSG
//CRASTRS DD DISP=SHR,DSN=FEK.#CUST.CRASTRS
//*
//SYSPROC DD DISP=SHR,DSN=ISP.SISPCLIB
//          DD DISP=SHR,DSN=FEK.SFEKPROC
//ISPEXEC DD DISP=SHR,DSN=ISP.SISPEXEC
//ISPLIB DD DISP=SHR,DSN=ISP.SISPMENU
//ISPPLIB DD DISP=SHR,DSN=ISP.SISPPENU
//ISPSLIB DD DISP=SHR,DSN=ISP.SISPSENU
//ISPTLIB DD DISP=SHR,DSN=ISP.SISPTENU
//ISPTL0 DD DISP=(NEW,DELETE,DELETE),UNIT=SYSALLDA,
//          SPACE=(TRK,(1,1)),LRECL=80,RECFM=FB
//ISPTL1 DD DISP=(NEW,DELETE,DELETE),UNIT=SYSALLDA,
//          SPACE=(TRK,(1,1)),LRECL=80,RECFM=FB
//ISPPROF DD DISP=(NEW,DELETE,DELETE),UNIT=SYSALLDA,
//          SPACE=(TRK,(1,1,5)),LRECL=80,RECFM=FB
//*
//CARMALOG DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD DUMMY
//*
//CONLIB DD DISP=SHR,DSN=CA.NDVR.CONLIB
//JCLOUT DD SYSOUT=(A,INTRDR),DCB=(LRECL=80,RECFM=F,BLKSIZE=80)
//EXT1ELM DD DISP=(NEW,DELETE),UNIT=SYSALLDA,
//          RECFM=VB,LRECL=4096,BLKSIZE=27998,SPACE=(TRK,(5,5))
//EXT1DEP DD DISP=(NEW,DELETE),UNIT=SYSALLDA,
//          RECFM=VB,LRECL=4096,BLKSIZE=27998,SPACE=(TRK,(5,5))
//MSG3FILE DD DISP=(NEW,DELETE),UNIT=SYSALLDA,
//          RECFM=FB,LRECL=133,BLKSIZE=27930,SPACE=(TRK,(5,5))
//C1MSG51 DD DISP=(NEW,DELETE),UNIT=SYSALLDA,
//          RECFM=FB,LRECL=133,BLKSIZE=27930,SPACE=(TRK,(5,5))
//C1EXMSG5 DD DISP=(NEW,DELETE),UNIT=SYSALLDA,
//          RECFM=FB,LRECL=133,BLKSIZE=27930,SPACE=(TRK,(5,5))
//TYPEMAP DD DISP=SHR,DSN=FEK.#CUST.PARMLIB(CRATMAP)
//SHOWVIEW DD DISP=SHR,DSN=FEK.#CUST.PARMLIB(CRASHOW)
$$
EXIT CODE(0)

```

Figure 18. Figure x2. CRASUBCA - CA Endeavor® SCM RAM startup using batch submit

Note:

- You can add your own DD statements and data set concatenations to customize the CARMA TSO environment, thus mimicking a TSO logon procedure.
- You can optionally change CARMA's timeout value by modifying the PROC 1 PORT TIMEOUT(420) line in the 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).
- Details of the CARMA startup process are shown in rsecomm.log. Refer to "(Optional) RSE tracing" on page 88 for more information on setting the detail level of rsecomm.log.
- Changes are in effect for all CARMA servers started after the update.

CA Endeavor® SCM RAM startup using CRASTART

Do not execute this step if you use the batch submit method to start the CARMA server with the CA Endeavor® SCM RAM.

Developer for System z can use the CRASTART CARMA server startup method to start the CA Endeavor® SCM RAM. The method uses CRASTART to start the CARMA server as a subtask within RSE..

Refer to “(Optional) Alternative CARMA server startup using CRASTART” on page 50 for more information on the CRASTART startup method.

Note: Details of the CARMA startup process are shown in `rsecomm.log`. Refer to “(Optional) RSE tracing” on page 88 for more information on setting the detail level of `rsecomm.log`.

Adjust CRASRV.properties

RSE server uses the settings in `/etc/rdz/CRASRV.properties` to start and connect to a CARMA server, as documented in “RSE interface to CARMA” on page 47. You can edit the file with the TSO **OEDIT** command. Note that RSE must be restarted for the changes to take effect.

Change the value of the `clist.dsname` directive to `*CRASTART` and provide the correct values for the `crastart.*` directives, as shown in the following example. Refer to “RSE interface to CARMA” on page 47 for more information on the different directives.

```
port.start=5227
port.range=100
startup.script.name=/usr/lpp/rdz/bin/carma.startup.rex
clist.dsname=*CRASTART
crastart.stub=/usr/lpp/rdz/bin/CRASTART
crastart.configuration.file=/etc/rdz/crastart.endevor.conf
crastart.syslog=Partial
crastart.timeout=420
#crastart.steplib=FKE.SFEKLPA
#crastart.tasklib=TASKLIB
```

Figure 19. Figure x3. `CRASRV.properties` - CA Endeavor® SCM RAM startup using CRASTART

Note: System abend 522 for module CRASERV will occur if the `JWT` parameter in the `SMFPRMxx` parmlib member is set to a value lower than the time out value in `CRASRV.properties`. This does not impact CARMA operations, because the server is restarted automatically if needed.

Adjust crastart.endevor.conf

CRASTART uses the definitions in `crastart.endevor.conf` to create a valid (TSO/ISPF) environment to invoke CA Endeavor® SCM. Developer for System z uses this environment to run the CA Endeavor® SCM RAM.

`crastart.endevor.conf` is located in `/etc/rdz/`, unless you specified a different location when you customized and submitted job `FKE.SFEKSAMP(FEKSETUP)`. See “Customization setup” on page 13 for more details. You can edit the file with the TSO **OEDIT** command.

Note: Changes are in effect for all CARMA servers started after the update.

```

| TASKLIB = FEK.SFEKLOAD,CA.NDVR.AUTHLIB,CA.NDVRU.AUTHLIB
| CRADEF = FEK.#CUST.CRADEF
| CRAMSG = FEK.#CUST.CRAMSG
| CRASTRS = FEK.#CUST.CRASTRS
|
| SYSPROC = ISP.SISPCLIB,FEK.SFEKPROC
| SYSEXEC = ISP.SISPEXEC
| ISPLIB = ISP.SISPMENU
| ISPLIB = ISP.SISPPENU
| ISPLIB = ISP.SISPSENU
| -COMMAND=ALLOC FI(ISPCTL0) NEW DELETE DSORG(PS) RECFM(F,B) LRECL(80)
| BLKSIZE(6160) SPACE(2,2) TRACKS UNIT(SYSALLDA)
| -COMMAND=ALLOC FI(ISPCTL1) NEW DELETE DSORG(PS) RECFM(F,B) LRECL(80)
| BLKSIZE(6160) SPACE(2,2) TRACKS UNIT(SYSALLDA)
| -COMMAND=ALLOC FI(ISPPROF) NEW DELETE DSORG(PO) DIR(5) RECFM(F,B) LRECL(80)
| BLKSIZE(6160) SPACE(2,2) TRACKS UNIT(SYSALLDA)
| ISPTLIB = -ISPPROF,ISP.SISPTENU
| ISPTABL = -ISPPROF
|
| CARMALOG= SYSOUT(H)
| SYSPRINT= SYSOUT(H)
| SYSTSPRT= SYSOUT(H)
| SYSTSIN = DUMMY
|
| TYPEMAP = FEK.#CUST.PARMLIB(CRATMAP)
| SHOWVIEW= FEK.#CUST.PARMLIB(CRASHOW)
| CONLIB = CA.NDVR.CONLIB
| -COMMAND=ALLOC FI(JCLOUT) SYSOUT(A) WRITER(INTRDR) RECFM(F) LRECL(80)
| BLKSIZE(80)
| -COMMAND=ALLOC FI(EXT1ELM) NEW DELETE DSORG(PS) RECFM(V,B) LRECL(4096)
| BLKSIZE(27998) SPACE(5,5) TRACKS UNIT(SYSALLDA)
| -COMMAND=ALLOC FI(EXTIDEP) NEW DELETE DSORG(PS) RECFM(V,B) LRECL(4096)
| BLKSIZE(27998) SPACE(5,5) TRACKS UNIT(SYSALLDA)
| -COMMAND=ALLOC FI(MSG3FILE) NEW DELETE DSORG(PS) RECFM(F,B) LRECL(133)
| BLKSIZE(27930) SPACE(5,5) TRACKS UNIT(SYSALLDA)
| -COMMAND=ALLOC FI(C1EXMSG5) NEW DELETE DSORG(PS) RECFM(F,B) LRECL(133)
| BLKSIZE(27930) SPACE(5,5) TRACKS UNIT(SYSALLDA)
| -COMMAND=ALLOC FI(C1MSG51) NEW DELETE DSORG(PS) RECFM(F,B) LRECL(133)
| BLKSIZE(27930) SPACE(5,5) TRACKS UNIT(SYSALLDA)
|
| PROGRAM=IKJEFT01 %CRANDVRA NDVRC1 PGM(CRASERV) PARM(&CRAPRM1.
| &CRAPRM2.)

```

Figure 20. crastart.conf - CA Endeavor® SCM RAM startup using CRASTART

(Optional) Customize CRANDVRA

Both the batch submit and the CRASTART startup method invoke REXX exec CRANDVRA to allocate user-specific data sets used by CA Endeavor® SCM RAM.

DD	Data set name	Type
DEPEND	&SYSPREF..&SYSUID.. &SYSNAME..CRA\$NDVR.DEPEND	Permanent
BROWSE	&SYSPREF..&SYSUID.. &SYSNAME..CRA\$NDVR.BROWSE	Temporary
C1PRINT	&SYSPREF..&SYSUID.. &SYSNAME..CRA\$NDVR.LISTING	Temporary

You can customize a copy of this allocation REXX exec if certain defaults, such as the data set name, do not match your site standards. CRANDVRA is located in FEK.SFEKPROC, unless you used a different high level qualifier during the SMP/E install of Developer for System z.

Refer to the documentation within CRANDVRA for customization instructions.

Note: You should copy the sample allocation REXX to a new data set and customize this copy to avoid overwriting it when applying maintenance. When you do this, you must update the reference to SFEKPROC in the SYSEXEC DD of your chosen CARMA startup method to match your new data set name.

(Optional) Customize the CA Endeavor® SCM RAM

The following CARMA components can be customized, regardless of the chosen startup method. The sample members referenced below are located in FEK.#CUST.PARMLIB, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details.

1. (Optional) Customize FEK.#CUST.PARMLIB(CRASHOW). Refer to the documentation within CRASHOW for customization instructions. CRASHOW defines default filters for CA Endeavor® SCM environments, systems, and so forth.
2. (Optional) Customize FEK.#CUST.PARMLIB(CRATMAP). Refer to the documentation within CRATMAP for customization instructions. CRATMAP overrides CA Endeavor® SCM type to file extension mappings.

(Optional) Supporting multiple RAMs

CARMA allows that multiple RAMs are defined and can run them concurrently. However, since there is only one CARMA server active per user, even when there are multiple RAMs, some configuration changes might be required to make this setup work.

RAMs are defined by a RAM developer in the CARMA configuration VSAM data set, CRADEF. During startup, the CARMA server, CRASERV, will identify all defined RAMs and present the information to the CARMA client. The user can then select one or more RAMs, which will be loaded into the CARMA server.

Since RAMs are active as plug-ins of the CARMA server, you must ensure that all prerequisites (such as data set allocations) for each of the RAMs are available in the address space of the CARMA server. This might require changes to the CARMA configuration samples, such as CRASUBMT or crastart.conf, which are shipped with Developer for System z.

Example

In the following example, you start from an existing setup with the CA Endeavor® SCM RAM, using the CRASTART startup method, and add the sample PDS RAM.

Definitions for the CA Endeavor® SCM RAM:

- FEK.SFEKVSM2(CRA0VCAD) - CRADEF definitions
- FEK.SFEKVSM2(CRA0VCAS) - CRASTRS definitions
- /etc/rdz/crastart.endeavor.conf - CRASTART configuration file

Definitions for the PDS RAM:

- FEK.SFEKVSM2(CRA0VDEF) - CRADEF definitions
- FEK.SFEKVSM2(CRA0VSTR) - CRASTRS definitions
- FEK.#CUST.CRARAM1 - CRARAM1 definitions

The process starts with a RAM developer gathering the data and information needed by the system programmer to complete the setup.

1. Extract the data specific for the PDS RAM from the SFEKVSM2 members (these members hold definitions for all sample RAMs, not just the PDS RAM).
2. Merge this data with the CA Endeavor[®] SCM RAM SFEKVSM2 members.
3. Create a list of PDS RAM specific prerequisites:
 - DD CRARAM1, linked to FEK.#CUST.CRARAM1
 - TSO environment

The system programmer then uses this data to create the updated CARMA VSAM data sets and uses the prerequisite information to create a CRASTART configuration file that is capable of supporting both RAMs.

1. Use the combined data as input for the CRA\$VDEF and CRA\$VSTR jobs to create the updated CARMA configuration and custom information VSAM data sets, CRADEF and CRASTRS.
2. Add a CRARAM1 definition to crastart.endevor.conf:

```
CRARAM1 = FEK.#CUST.CRARAM1
```

3. Verify the PROGRAM statement in crastart.endevor.conf to ensure it is capable of providing the environment needed by both RAMs:

```
PROGRAM=IKJEFT01 %CRANDVRA NDVRC1 PGM(CRASERV)
      PARM(&CRAPRM1. &CRAPRM2.)
```

- IKJEFT01: TSO, used to allow certain authorized calls in a non-authorized environment, and used as environment to run the CA Endeavor[®] SCM RAM pre-allocation exec.
- %CRANDVRA: CA Endeavor[®] SCM RAM pre-allocation exec (located in FEK.SFEKPROC), that allocates temporary (and permanent) user-specific working data sets.
- NDVRC1: CA Endeavor[®] back-end, which has a built in mechanism to execute TSO and ISPF commands.
- PGM(CRASERV): command to start a CARMA server, in ISPF command format
- PARM(&CRAPRM1. &CRAPRM2.): parameters for CRASERV, in ISPF command format. &CRAPRM1 is the port to be used and &CRAPRM2 is the timeout value.

The CA Endeavor[®] SCM RAM is active in an ISPF environment, which implies that the TSO environment required by the PDS RAM is also available.

(Optional) IRXJCL versus CRAXJCL

If the CARMA server is started using TSO (IKJEFTxx), you may experience problems if your RAMs call services which in turn call the IRXJCL REXX batch interface. The problem can occur when the processors called by the RAM previously ran either without TSO, or only in online TSO and dynamically allocates DD SYSTSIN or SYSTSPRT. A sample program, CRAXJCL, is provided to work around this problem.

Your processor might fail if it attempts to allocate SYSTSIN or SYSTSPRT (required for IRXJCL) because batch TSO (required for CARMA) already has those DD names allocated and open. The CRAXJCL replacement module attempts to allocate SYSTSIN and SYSTSPRT to DUMMY but ignores the errors which occur if the allocations fail.

This means that when your processors run in a CARMA environment started by TSO, the allocations to SYSTSIN and SYSTSPRT are the same as those used by CARMA. When the processors are run outside of TSO/CARMA, the SYSTSIN and SYSTSPRT allocations will be created by CRAXJCL. Therefore, your processors must not rely on the contents of the data set allocated to SYSTSIN.

It is assumed that calls to IRXJCL use the PARM field to pass the REXX name and startup parameters, as documented in *TSO/E REXX Reference* (SA22-7790). This means that SYSTSIN can safely be used by CARMA. Any output sent to SYSTSPRT by IRXJCL will end up in CARMA's log.

Processors that call the CRAXJCL replacement module should not attempt to allocate DD SYSTSIN or SYSTSPRT before calling CRAXJCL.

Create CRAXJCL

The CRAXJCL replacement module is shipped in source format because you will need to customize it to specify the specific allocations you want to use for SYSTSPRT. SYSTSIN should usually be allocated to a dummy data set.

Sample assembler source code and a sample compile/bind job are shipped as FEK.#CUST.ASM(CRAXJCL) and FEK.#CUST.JCL(CRA#CIRX) respectively, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See "Customization setup" on page 13 for more details.

Customize the CRAXJCL assembler source code per your needs, using the documentation within the member. Afterwards, customize and submit the CRA#CIRX JCL to create the CRAXJCL load module. Refer to the documentation within CRA#CIRX for customization instructions.

Chapter 4. (Optional) Application Deployment Manager

Developer for System z uses certain functions of Application Deployment Manager as a common deployment approach for various components. The customization steps listed in this chapter are required if your developers use any of the following functions:

- Enterprise Service Tools (EST)
- BMS Screen Designer
- MFS Screen Designer
- CICSTS Code Generation

Note: Enterprise Service Tools (EST) encompasses multiple tools, such as the Service Flow Modeler (SFM) and XML Services for the Enterprise (XSE).

Customizing Application Deployment Manager adds the CICS Resource Definition (CRD) server, which runs as a CICS application on z/OS to support the following functions:

- CICS resource queries
- CICS resource definition install and uninstall requests in both CICSplex SM and non-CICSplex SM environments
- Program and mapset phase-in requests
- Pipeline scan requests
- Manifest export, import, and update requests

CICS administrators can find more information on the CRD server in Chapter 15, “CICSTS considerations,” on page 231.

Requirements and checklist

You will need assistance of a CICS administrator, a TCP/IP administrator and a security administrator to complete this customization task, which requires the following resources or special customization tasks:

- TCP/IP port for external communication
- Update CICS region JCL
- Update CICS region CSD
- Define group to CICS region
- Security rule to allow administrators update to an Application Deployment Manager VSAM
- CICSTS security setup
- (Optional) Define CICS transaction names
- (Optional) Security rule to allow users update to an Application Deployment Manager VSAM

In order to start using Application Deployment Manager at your site, you must perform the following tasks. Unless otherwise indicated, all tasks are mandatory.

1. Create the CRD repository. For details, see “CRD repository” on page 66.
2. Choose the CICS interface (RESTful or Web Service) to be used. (The interfaces can co-exist). For details see “RESTful versus Web Service” on page 67.

3. If desired, do the RESTful specific customizations. For details see “CRD server using the RESTful interface” on page 67.
 - Define the CRD server to the CICS primary connection region
 - Optionally define the CRD server to CICS non-primary connection regions.
 - Optionally customize the CRD server transaction IDs.
4. If desired, do the Web Service specific customizations. For details see “CRD server using the Web Service interface” on page 68.
 - Add the (possibly customized) pipeline message handler to the CICS RPL concatenation.
 - Define the CRD server to the CICS primary connection region.
 - Optionally define the CRD server to CICS non-primary connection regions.
5. Optionally create the manifest repository. For details, see “(Optional) Manifest repository” on page 70.

CRD repository

Customize and submit job ADNVCRD to allocate and initialize the CRD repository VSAM data set. Refer to the documentation within the member for customization instructions.

ADNVCRD is located in FEK.#CUST.JCL, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details.

You should 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.

Note:

- An existing CRD server repository must be enlarged to enable the URIMAP support added to the Administrative utility in Developer for System z version 7.6.1. See “Administrative utility migration notes” on page 239 for more details.
- Unless notified otherwise, your current CRD server repository (holding your customized values) can be reused across Developer for System z releases.

Users require READ access to the CRD repository, CICS administrators require UPDATE access.

CICS administrative utility

Developer for System z provides the administrative utility to let CICS administrators provide the default values for CICS resource definitions. These defaults can be read-only, or can be editable by the application developer.

The administrative utility is invoked by sample job ADNJSPAU. The usage of this utility requires UPDATE access to the CRD repository.

ADNJSPAU is located in FEK.#CUST.JCL, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details.

More information is available in Chapter 15, “CICSTS considerations,” on page 231.

RESTful versus Web Service

CICS Transaction Server provides in version 4.1 and higher support for an HTTP interface designed using Representational State Transfer (RESTful) principles. This RESTful interface is now the strategic CICSTS interface for use by client applications. The older Web Service interface has been stabilized, and enhancements will be for the RESTful interface only.

Application Deployment Manager follows this statement of direction and requires the RESTful CRD server for all services that are new to Developer for System version 7.6 or higher.

The RESTful and Web Service interfaces can be active concurrently in a single CICS region, if desired. In this case, there will be two CRD servers active in the region. Both servers will share the same CRD repository. Note that CICS will issue some warnings about duplicate definitions when the second interface is defined to the region.

CRD server using the RESTful interface

The information in this section describes how to define the CRD server that uses the RESTful interface to communicate with the Developer for System z client.

The RESTful and Web Service interfaces can be active concurrently in a single CICS region, if desired. In this case, there will be two CRD servers active in the region. Both servers will share the same CRD repository. Note that CICS will issue some warnings about duplicate definitions when the second interface is defined to the region.

CICS primary connection region

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

- Place the load modules FEK.SFEKLOAD(ADNCRD*, ADNANAL and ADNREST) 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 ADNCSDRS to update the CICS System Definition (CSD) for the CICS primary connection region. Refer to the documentation within the member for customization instructions.

ADNCSDRS is located in FEK.#CUST.JCL, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details.

- Use the appropriate CEDA command to install the Application Deployment Manager group for this region, for example:
CEDA INSTALL GROUP(ADNPCRGP)

CICS non-primary connection regions

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

Note: It is not necessary to perform these steps if CICSplex® SM Business Application Services (BAS) is used to manage your CICS resource definitions.

- Place the Application Deployment Manager load module FEK.SFEKLOAD(ADNCRD*) 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 ADNCSDAR to update the CSD for these non-primary, connection regions. Refer to the documentation within the member for customization instructions.

ADNCSDAR is located in FEK.#CUST.JCL, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details.

- Use the appropriate CEDA command to install the Application Deployment Manager group for these regions, for example:

```
CEDA INSTALL GROUP(ADNARRGP)
```

(Optional) Customize CRD server transaction IDs

Developer for System z supplies multiple transactions that are used by the CRD server when defining and inquiring CICS resources.

Table 11. Default CRD server transaction IDs

Transaction	Description
ADMS	For requests from the Manifest Processing tool to change CICS resources. 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.

You can change the transaction IDs to match your site standards by following these steps:

1. Customize and submit ADNTXNC to create load module ADNRCUST. Refer to the documentation within the member for customization instructions.
2. Place the resulting ADNRCUST load module in the CICS RPL concatenation (DD statement DFHRPL) of the CICS regions where the CRD server is defined.
3. Customize and submit ADNCSDTX to define ADNRCUST as program to the CICS regions where the CRD server is defined. Refer to the documentation within the member for customization instructions.

Note: The RESTful CRD server will always try to load the ADNRCUST load module. So you can get a small performance benefit by creating and defining the ADNRCUST load module, even if you do not change the transaction IDs.

CRD server using the Web Service interface

The information in this section describes how to define the CRD server that uses the Web Service interface to communicate with the Developer for System z client.

The RESTful and Web Service interfaces can be active concurrently in a single CICS region, if desired. In this case, there will be two CRD servers active in the region.

Both servers will share the same CRD repository. Note that CICS will issue some warnings about duplicate definitions when the second interface is defined to the region.

Pipeline message handler

The pipeline message handler (ADNTMSGH) is used for security by processing the user ID and password in the SOAP header. ADNTMSGH is referenced by the sample pipeline configuration file and must therefore be placed into the CICS RPL concatenation. Refer to Chapter 15, “CICSTS considerations,” on page 231 to learn more about the pipeline message handler and the required security setup.

Developer for System z supplies multiple transactions that are used by the CRD server when defining and inquiring CICS resources. These transaction IDs are set by ADNTMSGH, depending on the requested operation. Sample COBOL source code is provided to allow site-specific customizations to ADNTMSGH:

Table 12. Default CRD server transaction IDs

Transaction	Description
ADMS	For requests from the Manifest Processing tool to change CICS resources. 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.

Using the default:

- Place the FEK.SFEKLOAD(ADNTMSGH) 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 ADNTMSGH:

Sample members ADNMSGH* are located in FEK.#CUST.JCL and FEK.#CUST.COBOL, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details.

- Customize the sample Pipeline Message Handler (COBOL) source code, FEK.#CUST.COBOL(ADNMSGHS), to match your site’s standards.
- Customize and submit job FEK.#CUST.JCL(ADNMSGHC) to compile the customized ADNMSGHS source. Refer to the documentation within ADNMSGHC for customization instructions. Note that the resulting load module must be named ADNTMSGH.
- Place the resulting ADNTMSGH load module in the CICS RPL concatenation (DD statement DFHRPL) of the CICS primary connection region.

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

CICS primary connection region

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

- Place the load modules FEK.SFEKLOAD(ADNCRD*, ADNANAL and ADNREST) 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. Note that the pipeline message handler load module, ADNTMSGH, must also be placed in the RPL concatenation, as described in “Pipeline message handler” on page 69.

- Customize and submit job ADNCSDWS to update the CICS System Definition (CSD) for the CICS primary connection region. Refer to the documentation within the member for customization instructions. Note that the transaction IDs used in this job must match the ones used by the Pipeline message handler (which may have been customized).

ADNCSDWS is located in FEK.#CUST.JCL, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details.

- Use the appropriate CEDA command to install the Application Deployment Manager group for this region, for example:

```
CEDA INSTALL GROUP(ADNPCRGP)
```

CICS non-primary connection regions

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

Note: It is not necessary to perform these steps if CICSplex SM Business Application Services (BAS) is used to manage your CICS resource definitions.

- Place the Application Deployment Manager load modules FEK.SFEKLOAD(ADNCRD*) in the CICS RPL concatenation (DD statement DFHRPL) of these non-primary connection regions. You should do this by adding the installation data set to the concatenation so that applied maintenance is automatically available to CICS.
- Customize and submit job ADNCSDAR to update the CSD for these non-primary, connection regions. Refer to the documentation within the member for customization instructions.

ADNCSDAR is located in FEK.#CUST.JCL, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details.

- Use the appropriate CEDA command to install the Application Deployment Manager group for these regions, for example:

```
CEDA INSTALL GROUP(ADNARRGP)
```

(Optional) Manifest repository

Developer for System z allows clients to browse and optionally change manifests describing selected CICS resources. Depending on permissions set by the CICS administrator, changes can be done directly or exported to the manifest repository for further processing by a CICS administrator.

Note:

- This step is only required for customers that export manifests from Developer for System z to be processed by the Manifest Processing tool.
- The Manifest Processing tool is a plug-in for IBM CICS Explorer.

Customize and submit job ADNVMFST to allocate and initialize the manifest repository VSAM data set, and to define it to the CICS primary connection region. Refer to the documentation within the member for customization instructions. A

separate manifest repository must be created for each CICS primary connection region. All users need UPDATE access to the manifest repository.

ADNVMFST is located in FEK.#CUST.JCL, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details.

Chapter 5. (Optional) SCLM Developer Toolkit

SCLM Developer Toolkit 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 has an Eclipse-based plugin 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.

Requirements and checklist

You will need assistance of an SCLM administrator and optionally a security administrator to complete this customization task, which requires the following resources and/or special customization tasks:

- APF and LINKLIST updates
- Define SCLM language translators for JAVA/J2EE support
- Define SCLM types for JAVA/J2EE support
- (Optional) Security rule to allow users update to an SCLM VSAM
- (Optional) Installation of Ant

In order to start using SCLM Developer Toolkit at your site, you must perform the following tasks. Unless otherwise indicated, all tasks are mandatory.

1. Verify and adjust prerequisites and PARMLIB updates. For details, see “Prerequisites.”
2. Customize Developer for System z configuration files. For details see:
 - “ISPF.conf updates for SCLMDT” on page 74
 - “rsed.envvars updates for SCLMDT” on page 75
3. Optionally define long/short name translation support. For details, see “(Optional) Long/short name translation” on page 75.
4. Optionally install and customize Ant to use the JAVA/J2EE build support. For details, see “(Optional) Install and customize Ant” on page 78.
5. Update SCLM to define SCLMDT-specific parts. For details, see “SCLM updates for SCLMDT” on page 79.
6. Optionally set up automation to periodically clean up the SCLMDT work area. For details, see “Remove old files from WORKAREA” on page 80.

Prerequisites

Refer to Appendix E, “Requisites,” on page 305 for a list of required SCLM maintenance.

This appendix also documents the Ant specifications needed for JAVA/J2EE builds in SCLM Developer Toolkit.

Attention: SCLM Developer Toolkit requires the usage of ISPF's TSO/ISPF Client Gateway, which implies that z/OS 1.8 or higher is required.

As described in “PARMLIB changes” on page 14, SCLM Developer Toolkit requires additional customization of system settings. These changes include:

- (BPXPRMxx) Increase the maximum number of processes per z/OS UNIX user ID.
- (PROGxx) APF authorize SYS1.LINKLIB and the REXX runtime, REXX.V1R4M0.SEAGLPA or REXX.V1R4M0.SEAGALT.
- (PROGxx/LPALSTxx) Place ISP.SISPLPA, ISP.SISPLPAD, SYS1.LINKLIB and the REXX runtime in LINKLIST/LPALIB.

Also, SCLM Developer Toolkit uses SDSF or the TSO **OUTPUT** command to retrieve job completion status and job output. Both methods require some additional attention:

- SDSF must be ordered, installed, and configured separately. It also requires the usage of JES2.
- The default settings for the TSO **OUTPUT** command let a user retrieve job output that begins with his user ID only. If you want to use the **OUTPUT** facility fully, then the sample TSO/E exit IKJEFF53 might need to be modified so that a user can retrieve job output he owns, but that does not begin with his user ID. For more information about this exit, refer to *TSO/E Customization* (SA22-7783).

Users require READ, WRITE, and EXECUTE permission to the z/OS UNIX directories /tmp/ and /var/rdz/WORKAREA/. Directory WORKAREA/ is located in /var/rdz/, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details.

ISPF.conf updates for SCLMDT

SCLM Developer Toolkit uses the standard ISPF/SCLM skeletons, so ensure that skeleton library ISP.SISPLIB is allocated to the ISPSLIB concatenation in ISPF.conf. The usage of the ISP.SISPSENU data set is optional.

ISPF.conf is located in /etc/rdz/, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details. You can edit the file with the TSO **OEDIT** command.

Note: Changes are in effect for all clients connecting to the host after the update.

The following sample code shows the ISPF.conf file, which must be customized to match your system environment. Comment lines start with an asterisk (*). Add data sets to the concatenation on the same line and separate the names with a comma (.). See “ISPF.conf, ISPF’s TSO/ISPF Client Gateway configuration file” on page 42 for more details on customizing ISPF.conf.

```

* REQUIRED:
sysproc=ISP.SISPCLIB,FEK.SFEKPROC
ispmllib=ISP.SISPMENU
isptlib=ISP.SISPTENU
isplib=ISP.SISPPENU
ispslib=ISP.SISPSLIB

* OPTIONAL:
*allocjob = FEK.#CUST.CNTL(CRAISPRX)
*ISPF_timeout = 900

```

Figure 21. ISPF.conf updates for SCLMDT

Note:

- You can add your own DD-like statements and data set concatenations to customize the TSO environment, thus mimicking a TSO logon procedure. See Chapter 16, “Customizing the TSO environment,” on page 243 for more details.
- When you are doing batch builds, ensure that the customized version of the FLMLIBS skeleton is concatenated before the ISPF/SCLM skeleton library.

```
ispslib=hlq.USERSKEL,ISP.SISPSLIB
```

rsed.envvars updates for SCLMDT

SCLM Developer Toolkit uses some directives set in rsed.envvars to locate data sets and directories.

rsed.envvars is located in /etc/rdz/, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details. You can edit the file with the TSO **OEDIT** command.

Note: The RSED started task must be restarted to pick up any changes you make.

The following code sample shows the SCLMDT directives in rsed.envvars, which must be customized to match your system environment. See “rsed.envvars, RSE configuration file” on page 28 for more details on customizing rsed.envvars.

```

_SCLMDT_CONF_HOME=/var/rdz/sclmdt
#STEPLIB=$STEPLIB:FEK.SFEKAUTH:FEK.SFEKLOAD
#_SCLMDT_TRANSTABLE=FEK.#CUST.LSTRANS.FILE
#ANT_HOME=/usr/lpp/Apache/Ant/apache-ant-1.7.1
_SCLMDT_BASE_HOME=$RSE_HOME
_SCLMDT_WORK_HOME=$_CMDSERV_WORK_HOME
CGI_DWORK=$_SCLMDT_WORK_HOME

```

Figure 22. rsed.envvars updates for SCLMDT

(Optional) Long/short name translation

SCLM Developer Toolkit provides the ability to store long name files (which are files with names greater than 8 characters or in mixed case) into SCLM. This is achieved through the use of a VSAM file that contains the mapping of the long file name to the 8 character member name used in SCLM.

Note:

- For versions previous to z/OS 1.8, this facility is provided through a base ISPF/SCLM PTF that addresses APAR OA11426.

- The long/short name translation is also used by other SCLM-related products, such as IBM SCLM Administrator Toolkit.

Create LSTRANS.FILE, the long/short name translation VSAM

Customize and submit sample member FLM02LST in the ISPF sample library ISP.SISPSAMP, to create the long/short name translation VSAM. The configuration steps in this publication expect the VSAM to be named FEK.#CUST.LSTRANS.FILE, as shown in the following sample setup JCL.

```

//FLM02LST 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 VSAM.
/* 3. Change all references of FEK.#CUST.LSTRANS.FILE to
/*     match your naming convention for the SCLM translate VSAM.
/*
//CREATE EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE FEK.#CUST.LSTRANS.FILE
SET MAXCC=0
DEFINE CLUSTER(NAME(FEK.#CUST.LSTRANS.FILE) -
               VOLUMES(*****)) -
               RECORDSIZE(58 2048) -
               SHAREOPTIONS(3 3) -
               CYLINDERS(1 1) -
               KEYS(8 0) -
               INDEXED) -
DATA (NAME(FEK.#CUST.LSTRANS.FILE.DATA)) -
INDEX (NAME(FEK.#CUST.LSTRANS.FILE.INDEX))

/* DEFINE ALTERNATE INDEX WITH NONUNIQUE KEYS -> ESDS */

DEFINE ALTERNATEINDEX(-
               NAME(FEK.#CUST.LSTRANS.FILE.AIX) -
               RELATE(FEK.#CUST.LSTRANS.FILE) -
               RECORDSIZE(58 2048) -
               VOLUMES(*****)) -
               CYLINDERS(1 1) -
               KEYS(50 8) -
               UPGRADE -
               NONUNIQUEKEY) -
DATA (NAME(FEK.#CUST.LSTRANS.FILE.AIX.DATA)) -
INDEX (NAME(FEK.#CUST.LSTRANS.FILE.AIX.INDEX))

/*
/*
//PRIME EXEC PGM=IDCAMS,COND=(0,LT)
//SYSPRINT DD SYSOUT=*
//INITREC DD *
INITREC1
/*
//SYSIN DD *
REPRO INFILE(INITREC) -
      OUTDATASET(FEK.#CUST.LSTRANS.FILE)
IF LASTCC = 4 THEN SET MAXCC=0

BLDINDEX IDS(FEK.#CUST.LSTRANS.FILE) -
          ODS(FEK.#CUST.LSTRANS.FILE.AIX)

IF LASTCC = 0 THEN -
  DEFINE PATH (NAME(FEK.#CUST.LSTRANS.FILE.PATH) -
              PATHENTRY (FEK.#CUST.LSTRANS.FILE.AIX))
/*

```

Figure 23. FLM02LST - long/short name translation setup JCL

Note: Users need UPDATE authority to this VSAM data set, as described in Chapter 10, “Security considerations,” on page 147.

rsed.envvars updates for long/short name translation

Before using the long/short name translation, uncomment and set the `rsed.envvars` environment variable `_SCLMDT_TRANTABLE` to match the name of the long/short name translation VSAM.

`rsed.envvars` is located in `/etc/rdz/`, unless you specified a different location when you customized and submitted job `FEK.SFEKSAMP(FEKSETUP)`. See “Customization setup” on page 13 for more details. You can edit the file with the TSO **OEDIT** command.

Note: The RSED started task must be restarted to pick up any changes you make.

(Optional) Install and customize Ant

This step is only required if you plan to use the JAVA/J2EE build support in SCLM.

Apache Ant is an open source Java build tool and can be downloaded from <http://ant.apache.org/>. Ant consists of text files and scripts which are distributed in ASCII format and thus require an ASCII/EBCDIC translation to run in z/OS UNIX.

Perform the following steps to implement Ant on z/OS, and to define it to Developer for System z:

- Download, in binary format, the latest Ant compressed file into the z/OS UNIX file system. It is recommended that you download the .zip version of ANT due to problems that might be encountered on z/OS when extracting versions of suffix format `tar.gz` or `tar.bz2`.
- Open a z/OS UNIX command-line session to continue the installation, for example with the **TSO OMVS** command.
- Make a home directory for the Ant install with the **mkdir -p /home-dir** command and make it your current directory with the **cd /home-dir** command.
- Use the JAR extract command **jar -xf apache-ant-1.7.1.zip** to extract the file to the current directory. A Java bin directory must be in your local z/OS UNIX PATH to use the **jar** command. Otherwise, fully qualify the command with the Java bin location (for example, **/usr/lpp/java/J5.0/bin/jar -xf apache-ant-1.7.1.zip**).
- Convert all Ant text files to EBCDIC by (optionally customizing and) executing sample script `/usr/lpp/rdz/samples/BWBTRANT`.

Note: Execute this script only once. Multiple runs will corrupt your Ant install.

- To check for successful translation, locate and browse a text file within the ANT directory, such as `apache-ant-1.7.1/README`. If the file is readable, then the translation was successful.
- Use the **chmod -R 755 *** command to enable all users to READ and EXECUTE files in the ANT directory.
- Before using Ant, set the `rsed.envvars` environment variables `JAVA_HOME` and `ANT_HOME`.
 - `JAVA_HOME` is required to point to the Java home directory, for example:
`JAVA_HOME=/usr/lpp/java/IBM/J5.0`
 - `ANT_HOME` is required to point to the Ant home directory, for example:
`ANT_HOME=/usr/lpp/Apache/Ant/apache-ant-1.7.1`

For example:

- TSO OMVS
- mkdir -p /usr/lpp/apache/Ant
- cd /usr/lpp/apache/Ant
- jar -xf /u/userid/apache-ant-1.7.1
- /usr/lpp/rdz/samples/BWBTRANT
- cat ./apache-ant-1.7.1/README
- chmod -R 755 *
- oedit /etc/rsed.envvars

To test that the Ant initialization has been successful:

- Add the Ant and Java bin directories to the environment variable PATH.

Example:

```
export PATH=/usr/lpp/apache/Ant/apache-ant-1.7.1/bin:$PATH
export PATH=/usr/lpp/java/IBM/J5.0/bin:$PATH
```

- Execute **ant -version** to display the version, if successfully installed.

Example:

```
ant -version
```

Note: Setting the PATH statement in this way is necessary for testing only, not for operational use.

SCLM updates for SCLMDT

SCLM itself also requires customization to work with SCLM Developer Toolkit. Refer to *IBM Rational Developer for System z SCLM Developer Toolkit Administrator's Guide* (SC23-9801) for more information on the required customization tasks:

- Define language translators for JAVA/J2EE support
- Define SCLM types for JAVA/J2EE support

To complete the customization and project definition tasks, the SCLM administrator needs to know several Developer for System z customizable values, as described in Table 13.

Table 13. SCLM administrator checklist

Description	<ul style="list-style-type: none"> • Default value • Where to find the answer 	Value
Developer for System z sample library	<ul style="list-style-type: none"> • FEK.SFEKSAMV • SMP/E installation 	
Developer for System z sample directory	<ul style="list-style-type: none"> • /usr/lpp/rdz/samples • SMP/E installation 	
Java bin directory	<ul style="list-style-type: none"> • /usr/lpp/java/J5.0/bin • rsed.envvars - \$JAVA_HOME/bin 	
Ant bin directory	<ul style="list-style-type: none"> • /usr/lpp/apache/Ant/apache-ant-1.7.1/bin • rsed.envvars - \$ANT_HOME/bin 	
WORKAREA home directory	<ul style="list-style-type: none"> • /var/rdz • rsed.envvars - \$_CMDSEV_CONF_HOME 	

Table 13. SCLM administrator checklist (continued)

Description	<ul style="list-style-type: none"> • Default value • Where to find the answer 	Value
SCLMDT project configuration home directory	<ul style="list-style-type: none"> • /var/rdz/sclmdt • rsed.envvars - \$_SCLMDT_CONF_HOME 	
Long/short name translation VSAM	<ul style="list-style-type: none"> • FEK.#CUST.LSTRANS.FILE • rsed.envvars - \$_SCLMDT_TRANTABLE 	

Remove old files from WORKAREA

SCLM Developer Toolkit and ISPF's TSO/ISPF Client Gateway share the same WORKAREA, which might need a periodical cleanup. Refer to "(Optional) WORKAREA cleanup" on page 98 for more information on this.

Chapter 6. (Optional) Other customization tasks

This section combines a variety of optional customization tasks. Follow the instructions in the appropriate section to configure the desired service.

- “(Optional) DB2 stored procedure”
- “(Optional) Enterprise Service Tools (EST) support” on page 83
- “(Optional) CICS bidirectional language support” on page 84
- “(Optional) Diagnostic IRZ error messages” on page 84
- “(Optional) RSE SSL encryption” on page 85
- “(Optional) RSE tracing” on page 88
- “(Optional) Host based property groups” on page 89
- “(Optional) Host based projects” on page 90
- “(Optional) File Manager integration” on page 91
- “(Optional) Uneditable characters” on page 92
- “(Optional) Using REXEC (or SSH)” on page 93
- “(Optional) APPC transaction for the TSO Commands service” on page 95
- “(Optional) WORKAREA cleanup” on page 98

(Optional) DB2 stored procedure

You will need the assistance of a WLM administrator and a DB2 administrator to complete this customization task, which requires the following resources or special customization tasks:

- WLM update
- New PROCLIB member
- DB2 update

Developer for System z provides a sample DB2 stored procedure (PL/I and COBOL Stored Procedure Builder) for building COBOL and PL/I Stored Procedures from within the Developer for System z client.

Note: Sample members ELAXM* are located in FEK.#CUST.JCL and FEK.#CUST.PROCLIB, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details.

Workload Manager (WLM) changes

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. Refer to *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.

PROCLIB changes

Customize the sample Stored Procedure task FEK.#CUST.PROCLIB(ELAXMSAM), as described within the member, and copy it to SYS1.PROCLIB. As shown in the following code sample, you have to provide the following:

- The name of the application environment defined in WLM for this Stored Procedure
- The DB2 subsystem name
- The high-level qualifier of various data sets

```
//ELAXMSAM PROC RGN=0M,
//          NUMTCB=1,
//          APPLENV=#w1mwd4z,
//          DB2SSN=#ssn,
//          DB2PRFX='DSN810',
//          COBPRFX='IGY.V3R4M0',
//          PLIPRFX='IBMZ.V3R6M0',
//          LIBPRFX='CEE',
//          LODPRFX='FEK'
//*
//DSNX9WLM EXEC PGM=DSNX9WLM,REGION=&RGN,TIME=NOLIMIT,DYNAMNBR=10,
//          PARM='&DB2SSN,&NUMTCB,&APPLENV'
//STEPLIB DD DISP=SHR,DSN=&DB2PRFX..SDSNEXIT
//          DD DISP=SHR,DSN=&DB2PRFX..SDSNLOAD
//          DD DISP=SHR,DSN=&LIBPRFX..SCEERUN
//          DD DISP=SHR,DSN=&COBPRFX..SIGYCOMP
//          DD DISP=SHR,DSN=&PLIPRFX..SIBMZCMP
//SYSEXEC DD DISP=SHR,DSN=&LODPRFX..SFEEKPROC
//SYSTSPRT DD SYSOUT=*
//CEEDUMP DD SYSOUT=*
//SYSABEND DD DUMMY
//SYSUT1 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT2 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT3 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT4 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT5 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT6 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT7 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//*
```

Figure 24. ELAXMSAM - DB2 stored procedure task

Note:

- The DB2 stored procedure uses REXX exec ELAXMREX, located in FEK.SFEKPROC. Do not change this location if you want possible SMP/E maintenance to be activated automatically.
- See Chapter 17, “Running multiple instances,” on page 249 if you want to rename members ELAXMSAM or ELAXMREX.

DB2 changes

Customize and submit sample member ELAXMJCL in data set FEK.#CUST.JCL to define the Stored Procedure to DB2. Refer to the documentation within the member for customization instructions.

```

//ELAXMJCL JOB <job parameters>
//JOBPROC JCLLIB ORDER=(#hlq.SDSNPROC)
//JOBLIB DD DISP=SHR,DSN=#hlq.SDSNEXIT
// DD DISP=SHR,DSN=#hlq.SDSNLOAD
//*
//RUNTIAD EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
DSN S(#ssn) R(1) T(1)
RUN PROGRAM(DSNTIAD) PLAN(DSNTIAD) -
LIB('#hlq.RUNLIB.LOAD')
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
CREATE PROCEDURE SYSPROC.ELAXMREX
( IN FUNCTION_REQUEST VARCHAR(20) CCSID EBCDIC
, IN SQL_ROUTINE_NAME VARCHAR(27) CCSID EBCDIC
, IN SQL_ROUTINE_SOURCE VARCHAR(32672) CCSID EBCDIC
, IN BIND_OPTIONS VARCHAR(1024) CCSID EBCDIC
, IN COMPILE_OPTIONS VARCHAR(255) CCSID EBCDIC
, IN PRECOMPILE_OPTIONS VARCHAR(255) CCSID EBCDIC
, IN PRELINK_OPTIONS VARCHAR(32672) CCSID EBCDIC
, IN LINK_OPTIONS VARCHAR(255) CCSID EBCDIC
, IN ALTER_STATEMENT VARCHAR(32672) CCSID EBCDIC
, IN SOURCE_DATASETNAME VARCHAR(80) CCSID EBCDIC
, IN BUILDOWNER VARCHAR(8) CCSID EBCDIC
, IN BUILDUTILITY VARCHAR(18) CCSID EBCDIC
, OUT RETURN_VALUE VARCHAR(255) CCSID EBCDIC )
PARAMETER STYLE GENERAL RESULT SETS 1
LANGUAGE REXX EXTERNAL NAME ELAXMREX
COLLID DSNREXCS WLM ENVIRONMENT ELAXMSAM
PROGRAM TYPE MAIN MODIFIES SQL DATA
STAY RESIDENT NO COMMIT ON RETURN NO
ASUTIME NO LIMIT SECURITY USER;

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

GRANT EXECUTE ON PROCEDURE SYSPROC.ELAXMREX TO PUBLIC;
//*
```

Figure 25. ELAXMJCL – DB2 stored procedure definition

Note: 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) Enterprise Service Tools (EST) support

This customization task does not require assistance, special resources, or special customization tasks.

The Developer for System z client has a code generation component called Enterprise Service Tools (EST). Depending on the type of code being generated, this code relies on functions provided by the Developer for System z host install. Making these host functions available is described in the following sections:

- Chapter 4, “(Optional) Application Deployment Manager,” on page 65
- “(Optional) CICS bidirectional language support” on page 84
- “(Optional) Diagnostic IRZ error messages” on page 84

Note: Enterprise Service Tools (EST) encompasses multiple tools, such as the Service Flow Modeler (SFM) and XML Services for the Enterprise (XSE).

(Optional) CICS bidirectional language support

You will need the assistance of a CICS administrator to complete this customization task, which requires the following resources or special customization tasks:

- Update CICS region JCL
 - Define a program to CICS
-

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.

Additionally, EST-generated code can support bidi transformation in environments other than CICS SFR (Service Flow Runtime). One example is 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, FEK.SFEKLOAD.

Perform the following tasks to activate CICS Bidirectional language support:

1. Place the FEK.SFEKLOAD load modules FEJBDCMP and FEJBDTRX in the CICS RPL concatenation (DD statement DFHRPL). You should do this by adding the installation data set to the concatenation so that applied maintenance is automatically available to CICS.

Note: If you do not concatenate the installation data set but copy the modules into a new or existing data set, keep in mind that those modules are DLLs and MUST reside in a PDSE library.

2. Define FEJBDCMP and FEJBDTRX as programs to CICS using the appropriate CEDA command, for example:

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

(Optional) Diagnostic IRZ error messages

This customization task does not require assistance, but does require the following resources or special customization tasks:

- LINKLIST update
 - Update CICS region JCL
-

The Developer for System z client has a code generation component called Enterprise Service Tools (EST). In order for code generated by EST to issue diagnostic error messages, all IRZ* and IIRZ* modules in the FEK.SFEKLOAD load library must be made available to the generated code. EST can generate code for the following environments:

- CICS
- IMS
- MVS batch

When the generated code is executed in a CICS transaction, then add all IRZ* and IIRZ* modules in FEK.SFEKLOAD to the DFHRPL DD of the CICS region. You should do this by adding the installation data set to the concatenation so that applied maintenance is automatically available to CICS.

In all other situations, make all IRZ* and IIRZ* modules in FEK.SFEKLOAD available either through STEPLIB or LINKLIST. You should do this by adding the installation data set to the concatenation so that applied maintenance is automatically available to CICS.

If you decide to use STEPLIB, you must define the modules not available through LINKLIST in the STEPLIB directive of the task that executes the code.

If the load modules are not available and an error is encountered by the generated code, then following message will be issued:

IRZ9999S Failed to retrieve the text of a Language Environment runtime message. Check that the Language Environment runtime message module for facility IRZ is installed in DFHRPL or STEPLIB.

(Optional) RSE SSL encryption

You will need assistance of a security administrator to complete this customization task, which requires the following resources or special customization tasks:

- LINKLIST update
 - Security rule to add program controlled data sets
 - (Optional) Security rule to add certificate for SSL
-

External (client-host) communication can be encrypted using SSL (Secure Socket Layer). This feature is disabled by default and is controlled by the settings in `ssl.properties`.

`ssl.properties` is located in `/etc/rdz/`, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP (FEKSETUP). See “Customization setup” on page 13 for more details. You can edit the file with the TSO **OEDIT** command. Note that RSE must be restarted for the changes to take effect.

The client communicates with RSE daemon during connection setup and with RSE server during the actual session. Both data streams are encrypted when SSL is enabled.

RSE daemon and RSE server support different mechanisms to store certificates due to architectural differences between the two. This implies that SSL definitions are

required for both RSE daemon and RSE server. A shared certificate can be used if RSE daemon and RSE server use the same certificate management method.

Table 14. SSL certificate storage mechanisms

Certificate storage	Created and managed by	RSE daemon	RSE server
key ring	SAF-compliant security product	supported	supported
key database	z/OS UNIX's gskkyman	supported	/
key store	Java's keytool	/	supported

Note:

- SAF-compliant key rings are the preferred method for managing certificates.
- SAF-compliant key rings can store the certificate's private key either in the security database or by using ICSF, the interface to System z cryptographic hardware. Access to ICSF is protected by profiles in the CSFSERV security class.

RSE daemon uses System SSL functions to manage SSL. This implies that SYS1.SIEALNKE must be program controlled by your security software and available to RSE via LINKLIST or the STEPLIB directive in rsed.envvars.

The following code sample shows the sample `ssl.properties` file, which must be customized to match your system environment. Comment lines start with a pound sign (#), when using a US code page. Data lines can only have a directive and its assigned value, comments are not allowed on the same line. Line continuations are not supported.

```
# ssl.properties - SSL configuration file
enable_ssl=false

# Daemon Properties

#daemon_keydb_file=
#daemon_keydb_password=
#daemon_key_label=

# Server Properties

#server_keystore_file=
#server_keystore_password=
#server_keystore_label=
#server_keystore_type=JCERACFKS
```

Figure 26. ssl.properties – SSL configuration file

The daemon and server properties only need to be set if you enable SSL. Refer to Appendix A, "Setting up SSL and X.509 authentication," on page 269 for more information on SSL setup.

enable_ssl

Enable or disable SSL communication. The default is false. The only valid options are true and false.

daemon_keydb_file

RACF (or similar security product) key ring name. Provide the key

database name if you used **gskkyman** to create a key database instead of using a key ring. Uncomment and customize this directive if SSL is enabled.

daemon_keydb_password

Leave commented out or blank if you use a key ring, otherwise provide the key database password. Uncomment and customize this directive if SSL is enabled and you are using a **gskkyman** key database.

daemon_key_label

The certificate label used in the key ring or key database, if it is not defined as the default one. Must be commented out if the default is used. Uncomment and customize this directive if SSL is enabled and you are not using the default security certificate.

server_keystore_file

Name of the key store created by Java's **keytool** command, or the RACF (or similar security product) key ring name if `server_keystore_type=JCERACFKS`. Uncomment and customize this directive if SSL is enabled.

server_keystore_password

Leave commented out or blank if you use a key ring, otherwise provide the key store password. Uncomment and customize this directive if SSL is enabled and you are using a **keytool** key store.

server_keystore_label

The certificate label used in the key ring or key store. The default is the first valid certificate encountered. Uncomment and customize this directive if SSL is enabled and you are not using the default security certificate.

server_keystore_type

Key store type. The default is JKS. Valid values are:

Table 15. Valid keystore types

Keyword	Key store type
JKS	Java key store
JCERACFKS	SAF-compliant key ring, where the certificate's private key is stored in the security database.
JCECCARACFKS	SAF-compliant key ring, where the certificate's private key is stored using ICSF, the interface to System z cryptographic hardware.

Note: At the time of publication, IBM z/OS Java requires an update of the `/usr/lpp/java/J5.0/lib/security/java.security` file to support JCECCARACFKS. The following line must be added:

```
security.provider.1=com.ibm.crypto.hwcca.provider.IBMJCECCA
```

The resulting file will look like this:

```
security.provider.1=com.ibm.crypto.hwcca.provider.IBMJCECCA
security.provider.2=com.ibm.jsse2.IBMJSSEProvider2
security.provider.3=com.ibm.crypto.provider.IBMJCE
security.provider.4=com.ibm.security.jgss.IBMJGSSProvider
security.provider.5=com.ibm.security.cert.IBMCertPath
security.provider.6=com.ibm.security.sasl.IBMSASL
```

(Optional) RSE tracing

This customization task does not require assistance, special resources, or special customization tasks.

Developer for System z supports different levels of tracing the internal program flow for problem solving purposes. RSE, and some of the services called by RSE, use the settings in `rsecomm.properties` to know the desired detail level in the output logs.

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

`rsecomm.properties` is located in `/etc/rdz/`, unless you specified a different location when you customized and submitted job `FEK.SFEKSAMP(FEKSETUP)`. See “Customization setup” on page 13 for more details. You can edit the file with the TSO **OEDIT** command.

The following code sample shows the `rsecomm.properties` file, which can be customized to match your tracing needs. Comment lines start with a pound sign (#), when using a US code page. Data lines can only have a directive and its assigned value, comments are not allowed on the same line. Line continuations are not supported.

```
# 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
debug_level=1

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

Figure 27. `rsecomm.properties` – Logging configuration file

server.version

Logging server version. The default is 5.0.0. Do not modify.

debug_level

Detail level for output logs. The default is 1 (log error and warning messages). Note that `debug_level` controls the detail level of multiple services (and thus multiple output files). Increasing the detail level will cause performance degradations and should only be done under the direction of the IBM support center. Refer to “RSE tracing” on page 135 for more information on which logs are controlled by this directive.

The valid values are the following:

0	Log error messages only.
1	Log error and warning messages.
2	Log error, warning, and informational messages.

Note: `debug_level` can be changed dynamically with the **modify rsecommlog** operator command, as described in Chapter 8, “Operator commands,” on page 115.

log_location

Output medium for RSE related logging. The default is `Log_To_File`. Do not change when using the RSE daemon connection method (default), unless directed by the IBM support center.

The valid values are the following:

Log_To_File	Send log messages to a separate file in the log output directory. <ul style="list-style-type: none"> RSE daemon: <code>rsedaemon.log</code> in <code>daemonlog</code> RSE thread pools: <code>rseserver.log</code> in <code>daemonlog</code> User: <code>rsecomm.log</code> in <code>userlog/.eclipse/RSE/\$LOGNAME</code>
Log_To_StdOut	Send log messages to stdout. <ul style="list-style-type: none"> RSE daemon: rerouted to DD STDOUT in the RSED started task RSE thread pools: undefined User: rerouted to <code>stdout.log</code> in <code>userlog/.eclipse/RSE/\$LOGNAME</code>

`daemonlog` is the value of the `daemon.log` directive in `rsd.envvars`. If the `daemon.log` directive is commented out or not present, the home path of the user ID assigned to the RSED started task is used. The home path is defined in the OMVS security segment of the user ID.

User-specific logs go to `userlog/.eclipse/RSE/$LOGNAME`, where `userlog` is the value of the `user.log` directive in `rsd.envvars`, and `$LOGNAME` is the logon user ID (uppercase). If the `user.log` directive is commented out or not present, the home path of the user is used. The home path is defined in the OMVS security segment of the user ID.

(Optional) Host based property groups

This customization task does not require assistance, special resources, or special customization tasks.

Developer for System z clients can define property groups which hold default values for various properties (for example, the COBOL compiler options to use when compiling COBOL source code). Developer for System z has some default values built in, but also allows defining custom, system-specific defaults.

The location of the custom property group and default value configuration files is defined in `propertiescfg.properties`, which is located in `/etc/rdz/`, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details. You can edit the file with the TSO **OEDIT** command. Note that RSE must be restarted for the changes to take effect.

The following code sample shows the `propertiescfg.properties` file, which must be customized to match your system environment. Comment lines start with a pound sign (#), when using a US code page. Data lines can only have a directive and its assigned value. Comments are not allowed on the same line. Line continuations are not supported.

```
#
# host based property groups - root configuration file
#
ENABLED=FALSE
RDZ-VERSION=7.5.0.0
PROPERTY-GROUP=/var/rdz/properties
DEFAULT-VALUES=/var/rdz/properties
```

Figure 28. propertiescfg.properties - Host-based property groups configuration file

ENABLED

Indicates whether Developer for System z will use the property group and default value configuration files. The default is FALSE. The only valid options are TRUE and FALSE.

RDZ-VERSION

Minimum Developer for System z client level to use host-based property groups. The default is 7.5.0.0. Do not modify.

PROPERTY-GROUP

The location of the property group configuration file. The default is /var/rdz/properties.

DEFAULT-VALUES

The location of the default value configuration file. The default is /var/rdz/properties.

Refer to the Developer for System z Information Center (<http://publib.boulder.ibm.com/infocenter/ratdevz/v7r6/index.jsp>) for more information on creating the property group configuration file (propertygroups.xml) and the default value configuration file (defaultvalues.xml).

(Optional) Host based projects

This customization task does not require assistance, special resources, or special customization tasks.

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 in /etc/rdz/, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See "Customization setup" on page 13 for more details. You can edit the file with the TSO OEDIT command. Note that RSE must be restarted for the changes to take effect.

The following code sample shows the projectcfg.properties file, which must be customized to match your system environment. Comment lines start with a pound sign (#), when using a US code page. Data lines can only have a directive and its assigned value. Comments are not allowed on the same line. Line continuations are not supported.

```
#
# 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/rdz/projects
```

Figure 29. *projectcfg.properties – Host-based projects configuration file*

WSED-VERSION

Minimum Developer for System z client level to use host-based projects.
The default is 7.0.0.0. Do not modify.

PROJECT-HOME

The base directory for the project definitions. The default is
/var/rdz/projects.

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

Refer to Developer for System z Information Center (<http://publib.boulder.ibm.com/infocenter/ratdevz/v7r6/index.jsp>) for more
information on host-based projects.

(Optional) File Manager integration

You will need the assistance of a security administrator to complete this customization task,
which requires the following resources or special customization tasks:

- Security rule to add program controlled data sets
-

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 and edit of MVS data
sets (including all types of VSAM), creating and editing MVS data set templates
(including dynamic templates), and advanced copy utilities are 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 Prerequisites*
(SC23-7659) 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.

Note that both Developer for System z and File Manager no longer support the
batch interface to access File Manager services. Usage of the File Manager listener
is now required.

Note: In addition to the normal listener setup tasks described in your File
Manager documentation, Developer for System z requires that the server's
STEPLIB data sets are program controlled.

The File Manager Integration definitions needed by Developer for System z are stored in `FMIEXT.properties`, which is located in `/etc/rdz/`, unless you specified a different location when you customized and submitted job `FEK.SFEKSAMP(FEKSETUP)`. See “Customization setup” on page 13 more details. You can edit the file with the TSO **OEDIT** command. Note that RSE must be restarted for the changes to take effect.

The following code sample shows the `FMIEXT.properties` file, which must be customized to match your system environment. Comment lines start with a pound sign (`#`), when using a US code page. Data lines can only have a directive and its assigned value. Comments are not allowed on the same line. Line continuations are not supported.

```
# File Manager Integration (FMI) Extension properties
#
enabled=false
fmlistenport=1960
```

Figure 30. FMIEXT.properties – File Manager configuration file

enabled

Indicates whether the File Manager listener is available on the same host system or not. The default value is `false`. The only values allowed are `true` and `false`.

fmlistenport

Port used by the File Manager listener. The default is 1960. Communication on this port is confined to your host machine.

(Optional) Uneditable characters

This customization task does not require assistance, special resources, or special customization tasks.

Some characters do not translate well between host code pages (EBCDIC based) and client code pages (ASCII based). The Developer for System z client editor uses the definitions in `uchars.settings` file to identify these uneditable characters. When opening a data set with a character identified in `uchars.settings`, the editor will enforce read-only mode, to avoid corrupting the data set when it is saved.

`uchars.settings` is located in `/etc/rdz/`, unless you specified a different location when you customized and submitted job `FEK.SFEKSAMP(FEKSETUP)`. See “Customization setup” on page 13 for more details. You can edit the file with the TSO **OEDIT** command. Note that RSE must be restarted for the changes to take effect. Also note that it is advised not to change this file, unless directed by IBM support center.

```
# uchars.settings - uneditable code points
#
*          *          0D 15 25;

# DBCS (Japanese, Korean & Chinese)
IBM-930    *          0D 15 1E 1F 25;
IBM-933    *          0D 15 1E 1F 25;
IBM-935    *          0D 15 1E 1F 25;
IBM-937    *          0D 15 1E 1F 25;
IBM-939    *          0D 15 1E 1F 25;
IBM-1390   *          0D 15 1E 1F 25;
IBM-1399   *          0D 15 1E 1F 25;
IBM-1364   *          0D 15 1E 1F 25;
IBM-1371   *          0D 15 1E 1F 25;
IBM-1388   *          0D 15 1E 1F 25;

# UNICODE
UTF-8      *          0D 0A;
UTF-16BE   *          0D 0A;
UTF-16LE   *          0D 0A;
UTF-16     *          0D 0A;
```

Figure 31. uchars.settings - Uneditable characters configuration file

The file consists of multiple entries in the following format:

```
HOST-CODEPAGE LOCAL-CODEPAGE HEX-CODEPOINTS ;
```

Where HEX-CODEPOINTS is a blank-delimited list of 2-digit hexadecimal code points which identify the uneditable characters. The list must end with a semicolon (;).

The following syntax rules apply:

- Comment lines start with a pound sign (#), when using a US code page.
- Data lines can only have data, comments are not allowed on the same line.
- An asterisk (*) can be used for HOST-CODEPAGE and/or LOCAL-CODEPAGE. It acts as a wildcard and represents all codepages.
- Specific entries take precedence over generic (wildcard) entries.
- "host-cp *" takes precedence if "host-cp *" and "* local-cp" are both specified and not overridden by "host-cp local-cp".
- If the same codepage pair is specified more than once, then the last entry will be used.

(Optional) Using REXEC (or SSH)

This customization task does not require assistance, special resources, or special customization tasks.

RExec (Remote Execution) is a TCP/IP service to let clients execute a command on the host. SSH (Secure Shell) is a similar service, but here all communication is encrypted using SSL (Secure Socket Layer). Developer for System z uses either service for doing remote (host-based) actions in z/OS UNIX subprojects.

Developer for System z can also be configured to use REXEC (or SSH) to start an RSE server on the host. Note, however, that each connection started this way will result in a separate RSE server, each using a fair amount of system resources. Therefore, this alternate connection method is only viable for a small number of connections.

Also, since the REXEC (or SSH) alternative connection method bypasses the RSE daemon, it does not have access to all host services described in this publication, such as single server processing and audit. Contact IBM support to learn if a specific host service is supported by the REXEC alternate connection method.

Note: Developer for System z uses the z/OS UNIX version of REXEC, not the TSO version.

Remote (host-based) actions for z/OS UNIX subprojects

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. Refer to “REXEC (or SSH) set up” to know which port is used.

Alternate RSE connection method

Developer for System z clients need to know two values to start an RSE connection through REXEC (or SSH), as follows:

- The directory where the server.zseries startup script is located.
server.zseries is located in /etc/rdz/, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details.

Note: To discourage the usage of REXEC (or SSH) as alternate logon method, server.zseries is no longer copied automatically to /etc/rdz. If you want to use this function, you must manually copy it from /usr/lpp/rdz/bin, as shown in the following sample command:

```
cp /usr/lpp/rdz/bin/server.zseries /etc/rdz
```

- The port that is being used.
Refer to “REXEC (or SSH) set up” to know which port is used.

REXEC (or SSH) set up

Communications Server IP Configuration Guide (SC31-8775) describes the steps required to set up REXEC (or SSH). Refer to Appendix C, “Setting up INETD,” on page 291 for Developer for System z specific setup considerations (there are no Developer for System z specific setup steps).

A common port used by REXEC is 512. To verify this, you can check /etc/inetd.conf and /etc/services to find the port number used.

- 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
- 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: /etc/inetd.conf and /etc/services can have different names. Refer to Appendix C, “Setting up INETD,” on page 291 for more information.

(Optional) APPC transaction for the TSO Commands service

You will need assistance of an APPC administrator and a WLM administrator to complete this customization task, which requires the following resources or special customization tasks:

- LINKLIST update
 - APPC transaction
 - WLM update
-

The TSO Commands service can be implemented as an APPC transaction program, FEKFRSRV. This transaction acts as a host server to execute TSO and ISPF commands that are issued from the workstation. APPC is not required on the workstation because the client communicates with FEKFRSRV through RSE. Each client can have an active connection to multiple hosts at the same time.

Note:

- By default, Developer for System z uses ISPF's TSO/ISPF Client Gateway to access the TSO Commands service.
- If you are unfamiliar with APPC, refer to Appendix D, "Setting up APPC," on page 299 before continuing with this section.
- RSE uses the TCP/IP REXX socket API to communicate with FEKFRSRV. This implies that the TCP/IP load library, default TCPIP.SEZALOAD, must be available either via LINKLIST or the STEPLIB directive in rsed.envvars.
- RSE must be restarted for the described changes to take effect.

Preparation

- The following tasks are a prerequisite and must be completed before configuring the TSO Commands Server. The mentioned publications describe these tasks.
 1. Install, configure, and start VTAM® on your z/OS system. Refer to *Communications Server IP SNA Network Implementation Guide* (SC31-8777) for more information.
 2. Install, configure, and start TCP/IP on your z/OS system. Refer to Appendix B, "Setting up TCP/IP," on page 283 for more information.
 3. Configure and start APPC and the APPC transaction scheduler (ASCH) subsystem. Refer to Appendix D, "Setting up APPC," on page 299 for more information.
- The following sample REXX 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.ICQCLIB') 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 32. REXX for APPC ISPF panels

Note: Be aware that you can deactivate the APPC transaction with this tool; the transaction is still there but will not 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 16. APPC transaction checklist

Expertise	Required information: • Default value • Where to find the answer	Value
APPC	Data set name of TPDATA • Default: SYS1.APPCTP • Value is listed in SYS1.PARMLIB(APPCPMxx)	
APPC	Transaction name to be used (may not exist) • Default: FEKFRSRV • Existing transactions can be queried by selecting "TP Profile Administration" in the APPC ISPF menu	
APPC	APPC transaction class to be used • Default: A • APPC classes are defined in SYS1.PARMLIB(ASCHPMxx)	
WLM/SRM	TSO performance group and domain • No IBM default (site-dependent)	
RACF	Every Developer for System z user has access to an OMVS segment (this is required) • No IBM default (site-dependent) • 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) • No IBM default (site-dependent) • 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.

Implementation

Note: Sample members FEKAPPC* are located in FEK.#CUST.JCL, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details.

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 FEK.#CUST.JCL(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 D, “Setting up APPC,” on page 299 for more information.
 - The APPC transaction class used must have enough APPC initiators to allow one initiator for each concurrent user of Developer for System z.
2. Define the APPC transaction that will act as a command server. You can use the sample JCL FEK.#CUST.JCL(FEKAPPCC) to define this transaction. Instructions on how to customize this JCL are located within the JCL.

Note:

- a. If you changed the transaction program name (default FEKFRSRV) in this step, the new name must be assigned to `_FEKFSCMD_TP_NAME_` in `rsed.envvars`, as described in “rsed.envvars, RSE configuration file” on page 28.
 - b. The APPC transaction uses the REXX exec FEKFRSRV, located in FEK.SFEKPROC. Do not change this location if you want possible SMP/E maintenance to be activated automatically.
 - c. Sample JCL is also provided to display, FEK.#CUST.JCL(FEKAPPCL), or delete, FEK.#CUST.JCL(FEKAPPCX), the transaction.
3. Enable RSE to use APPC by uncommenting the `RSE_JAVAOPTS="$_RSE_JAVAOPTS -DTSO_SERVER=APPC"` directive in `rsed.envvars`, as described in “rsed.envvars, RSE configuration file” on page 28.
 4. Control the dispatching priority of the 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.
 5. Define a default OMVS segment for the system or an individual one for each user of Developer for System z.

6. Give Developer for System z users READ access to FEK.SFEKPROC, the Developer for System z TSO executable library.

APPC usage considerations

- When using APPC for the TSO Commands service, Developer for System z is dependent 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, refer to Appendix B, “Setting up TCP/IP,” on page 283 and *TCPIP.DATA configuration statements* in the *Communications Server IP Configuration Reference* (SC31-8776). You can test your TCP/IP configuration by starting the RSE daemon with the IVP=IVP parameter or with the fekfivpt installation verification program (IVP), as documented in Chapter 7, “Installation verification,” on page 99.
- When using APPC for the TSO Commands service, Developer for System z requires an extra socket (TCP/IP port) for host-confined communication per opened MVS file. Any available port will be used. This port selection mechanism cannot be changed.
- When using APPC for the TSO Commands service, reading and writing an 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. See “Opening MVS data sets fails” on page 144.
- When using APPC for the TSO Commands service to avoid setting up ISPF’s TSO/ISPF Client Gateway, be aware that other services, such as SCLM Developer Toolkit, rely on the TSO/ISPF Client Gateway.
- Refer to Appendix D, “Setting up APPC,” on page 299 for general APPC usage considerations

(Optional) WORKAREA cleanup

This customization task does not require assistance, special resources, or special customization tasks.

ISPF’s TSO/ISPF Client Gateway and the SCLM Developer Toolkit function use the WORKAREA directory to store temporary work files, which are removed before the session is closed. However, temporary output is sometimes left behind, for example, if there is a communication error while processing. For this reason, it is recommended that you clear out the WORKAREA directory from time to time.

z/OS UNIX provides a shell script, skulker, that deletes files based upon the directory they are in and their age. Combined with the z/OS UNIX cron daemon, which runs commands at specified dates and times, you can set up an automated tool that periodically cleans out the WORKAREA directory. Refer to *UNIX System Services Command Reference* (SA22-7802) for more information on the skulker script and the cron daemon.

Note: WORKAREA is located in /var/rdz/, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details.

Chapter 7. Installation verification

After completing the product customization, you can use the Installation Verification Programs (IVPs) described in this chapter to verify the successful setup of key product components.

Verify started tasks

JMON, JES Job Monitor

Start the JMON started task (or user job). The startup information in DD STDOUT should end with the following message:

```
JM200I Server initialization complete.
```

If the job ends with return code 66, then FEK.SFEKAUTH is not APF authorized.

Note: Start JES Job Monitor before continuing with the other IVP tests.

LOCKD, Lock daemon

Start the LOCKD started task (or user job). The lock daemon issues the following console message upon successful startup:

```
FEK501I Lock daemon started, port=4036, cleanup interval=1440,  
log level=1
```

RSED, RSE daemon

Start the RSED started task (or user job) with the IVP=IVP parameter. With this parameter, the server will end after doing some installation verification tests. The output of these tests is available in DD STDOUT. In case of certain errors, data will also be available in DD STDERR. The STDOUT data should look like the following sample:

Note: Start the RSE daemon, without the IVP parameter, before continuing with the other IVP tests. RSE daemon issues the following console message upon successful startup:

```
FEK002I RseDaemon started. (port=4035)  
  
RSE daemon IVP test  
  
Wed Jul 2 17:11:52 2008 UTC  
uid=8(STCRSE) gid=1(STCGROUP)  
  
RSE daemon port is 4035  
RSE configuration files located in /etc/rdz  
  
-----  
current environment variables  
-----  
@="/usr/lpp/rdz/bin/rsed.sh" @[1]="4035" @[2]="/etc/rdz"  
CGI_DTCONF="/var/rdz/sc1mdt"  
CGI_DTWORK="/var/rdz"  
CGI_TRANTABLE="FEK.#CUST.LSTRANS.FILE"  
CLASSPATH=".:usr/lpp/rdz/lib:usr/lpp/rdz/lib/dstore_core.jar:usr/lpp/  
ERRNO="0"  
HOME="/tmp"  
IFS="
```

```

"
JAVA_HOME="/usr/lpp/java/J5.0"
JAVA_PROPAGATE="NO"
LANG="C"
LIBPATH=".:usr/lib:usr/lpp/java/J5.0/bin:usr/lpp/java/J5.0/bin/classi
LINENO="66"
LOGNAME="STCRSE"
MAILCHECK="600"
OLDPWD="/tmp"
OPTIND="1"
PATH=".:usr/lpp/java/J5.0/bin:usr/lpp/rdz/bin:usr/lpp/ispf/bin:bin:/
PPID="33554711"
PS1="\$ "
PS2="> "
PS3="#? "
PS4="+ "
PWD="/etc/rdz"
RANDOM="27298"
RSE_CFG="/etc/rdz"
RSE_HOME="/usr/lpp/rdz"
RSE_LIB="/usr/lpp/rdz/lib"
SECONDS="0"
SHELL="/bin/sh"
STEPLIB="NONE"
TZ="EST5EDT"
_BPX_SHAREAS="YES"
_BPX_SPAWN_SCRIPT="YES"
_CEE_DMPTARG="/tmp"
_CEE_RUNOPTS="ALL31(ON) HEAP(32M,32K,ANYWHERE,KEEP,,) TRAP(ON)"
_CMDSERV_BASE_HOME="/usr/lpp/ispf"
_CMDSERV_CONF_HOME="/etc/rdz"
_CMDSERV_WORK_HOME="/var/rdz"
_RSE_CMDSERV_OPTS="&SESSION=SPAWN"
_RSE_DAEMON_CLASS="com.ibm.etools.zos.server.RseDaemon"
_RSE_DAEMON_IVP_TEST="1"
_RSE_DAEMON_PORT="4035"
_RSE_JAVAOPTS=" -DISPF_OPTS='&SESSION=SPAWN' -DA_PLUGIN_PATH=/usr/lpp/rd
_RSE_POOL_SERVER_CLASS="com.ibm.etools.zos.server.ThreadPoolProcess"
_RSE_PWD="/tmp"
_RSE_SERVER_CLASS="org.eclipse.dstore.core.server.Server"
_RSE_SERVER_TIMEOUT="120000"
_SCLMDT_BASE_HOME="/usr/lpp/rdz"
_SCLMDT_CONF_HOME="/var/rdz/sclmdt"
_SCLMDT_TRANTABLE="FEK.#CUST.LSTRANS.FILE"
_SCLMDT_WORK_HOME="/var/rdz"
_SCLM_BASE="/var/rdz/WORKAREA"
_SCLM_BWBCALL="/usr/lpp/rdz/bin/BWBCALL"
_SCLM_DWGET="/var/rdz/WORKAREA"
_SCLM_DWTRANSFER="/var/rdz/WORKAREA"
_SCLM_J2EPUT="/var/rdz/WORKAREA"

```

```

-----
java startup test...
-----

```

```

java version "1.5.0"
Java(TM) 2 Runtime Environment, Standard Edition (build pmz31dev-2008031
IBM J9 VM (build 2.3, J2RE 1.5.0 IBM J9 2.3 z/OS s390-31 j9vmmz3123-2008
J9VM - 20080314_17962_bHdSMr
JIT - 20080130_0718ifx2_r8
GC - 200802_08)
JCL - 20080314

```

```

-----
TCP/IP IVP test...
-----

```

```

Wed Jul 2 13:11:54 EDT 2008

```

```

uid=8(STCRSE) gid=1(STCGRP)
using /etc/rdz/rsed.envvars

-----
TCP/IP resolver configuration (z/OS UNIX search order):
-----
Resolver Trace Initialization Complete -> 2008/07/02 13:11:54.745964

res_init Resolver values:
Global Tcp/Ip Dataset = None
Default Tcp/Ip Dataset = None
Local Tcp/Ip Dataset = /etc/resolv.conf
Translation Table = Default
UserId/JobName = STCRSE
Caller API = LE C Sockets
Caller Mode = EBCDIC
(L) DataSetPrefix = TCPIP
(L) HostName = CDFMVS08
(L) TcpIpJobName = TCPIP
(L) DomainOrigin = RALEIGH.IBM.COM
(L) NameServer = 9.42.206.2
                  9.42.206.3
(L) NsPortAddr = 53          (L) ResolverTimeout = 10
(L) ResolveVia = UDP        (L) ResolverUdpRetries = 1
(*) Options NDots = 1
(*) SockNoTestStor
(*) AlwaysWto = NO          (L) MessageCase = MIXED
(*) LookUp = DNS LOCAL

res_init Succeeded
res_init Started: 2008/07/02 13:11:54.755363
res_init Ended: 2008/07/02 13:11:54.755371
*****
MVS TCP/IP NETSTAT CS V1R9      TCPIP Name: TCPIP      13:11:54
Tcpip started at 01:28:36 on 06/23/2008 with IPv6 enabled

-----
host IP address:
-----
hostName=CDFMVS08
hostAddr=9.42.112.75
bindAddr=9.42.112.75
localAddr=9.42.112.75

Success, addresses match

-----
PassTicket IVP test...
-----

Success, PassTicket IVP finished normally

-----
RSE daemon IVP ended

```

Verify services

The Developer for System z installation provides several Installation Verification Programs (IVP) for the basic and optional services. The IVP scripts are located in the installation directory, default `/usr/lpp/rdz/bin/`.

Table 17. IVPs for services

fekfivpa	“(Optional) TSO Commands service connection using APPC” on page 107
fekfivpd	“RSE daemon connection” on page 104

Table 17. IVPs for services (continued)

fekfivpi	"ISPF's TSO/ISPF Client Gateway connection" on page 106
fekfivpj	"JES Job Monitor connection" on page 105
fekfivpl	"Lock daemon connection" on page 105
fekfivpr	"(Optional) REXEC connection" on page 109
fekfivps	"(Optional) SCLMDT connection" on page 108
fekfivpt	"TCP/IP setup" on page 103
fekfivpz	"(Optional) REXEC/SSH shell script" on page 110

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.

A large region size is required for the user ID that executes the IVPs, because functions such as Java, which require a lot of memory, will be executed. You should set the region size to 131072 kilobytes (128 megabytes) or higher.

The following sample error is a clear indication of an insufficient region size. (But other errors can occur, too. For example, Java may fail to start.)

```
CEE5213S The signal SIGPIPE was received.
%z/OS UNIX command%: command was killed by signal number 13
  %line-number% ** %REXX command%
    +++ RC(137) +++
```

Note: The Developer for System z started tasks must be active before starting the IVP test.

IVP initialization

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

```
$ pwd
/u/userid
$ cd /etc/rdz
$ ./ivpinit
RSE configuration files located in /etc/rdz --default
added /usr/lpp/rdz/bin to PATH
```

The first "." (dot) in **./ivpinit** 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 **./ivpinit** is NOT executed before the **fekfivp*** scripts, the path to these scripts must be specified when calling them, as in the following sample:

```
/usr/lpp/rdz/bin/fekfivpr 512 USERID
```

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

- 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):

```
$ EXPORT STEPLIB=$STEPLIB:TCPIP.SEZALOAD
```

Note that adding a non-APF authorized library to an existing STEPLIB removes the APF authorization for existing STEPLIB data sets.

Also note that if CEE.SCEELKED is in LINKLIST or STEPLIB, TCPIP.SEZALOAD must be placed before CEE.SCEELKED. Failure to do so will result in a 0C1 system abend for the TCP/IP REXX socket calls.

For information on diagnosing RSE connection problems, see Chapter 9, “Troubleshooting configuration problems,” on page 127 or the Technotes on the Developer for System z Support Page <http://www-306.ibm.com/software/awdtools/rdz/support/>.

Port availability

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

IPv4

```
$ netstat
MVS TCP/IP NETSTAT CS VxRy    TCPIP Name: TCPIP      13:57:36
User Id Conn      Local Socket      Foreign Socket      State
-----
RSED      0000004B 0.0.0.0..4035     0.0.0.0..0         Listen
LOCKD     0000004C 0.0.0.0..4036     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 VxRy    TCPIP Name: TCPIP      14:03:35
User Id Conn      State
-----
RSED      0000004B Listen
Local Socket: 0.0.0.0..4035
Foreign Socket: 0.0.0.0..0
LOCKD     0000004C Listen
Local Socket: 0.0.0.0..4036
Foreign Socket: 0.0.0.0..0
JMON      00000037 Listen
Local Socket: 0.0.0.0..6715
Foreign Socket: 0.0.0.0..0
```

TCP/IP setup

When using APPC for the TSO Commands service, Developer for System z is dependent 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 B, “Setting up TCP/IP,” on page 283 for more information on TCP/IP and Resolver setup. Verify the current settings by executing the following command:

```
fekfivpt
```

Note: This IVP issues the TCP/IP **netstat** command, which might be protected against execution by your security software.

The command should return an output like that in this sample (\$ is the z/OS UNIX prompt):

```
$ fekfivpt
```

```
Wed Jul  2 13:11:54 EDT 2008
uid=1(USERID) gid=0(GROUP)
using /etc/rdz/rsed.envvars
```

```
current address space size limit is 1914675200 (1826.0 MB)
maximum address space size limit is 2147483647 (2048.0 MB)
```

```
-----
TCP/IP resolver configuration (z/OS UNIX search order):
-----
```

```
Resolver Trace Initialization Complete -> 2008/07/02 13:11:54.745964
```

```
res_init Resolver values:
```

```
Global Tcp/Ip Dataset = None
Default Tcp/Ip Dataset = None
Local Tcp/Ip Dataset  = /etc/resolv.conf
Translation Table      = Default
UserId/JobName         = USERID
Caller API             = LE C Sockets
Caller Mode            = EBCDIC
(L) DataSetPrefix     = TCPIP
(L) HostName           = CDFMVS08
(L) TcpIpJobName       = TCPIP
(L) DomainOrigin      = RALEIGH.IBM.COM
(L) NameServer         = 9.42.206.2
                      = 9.42.206.3
(L) NsPortAddr        = 53
(L) ResolveVia         = UDP
(L) ResolverTimeout   = 10
(L) ResolverUdpRetries = 1
(*) Options NDots      = 1
(*) SockNoTestStor     = 1
(*) AlwaysWto          = NO
(L) MessageCase        = MIXED
(*) LookUp             = DNS LOCAL
```

```
res_init Succeeded
```

```
res_init Started: 2008/07/02 13:11:54.755363
```

```
res_init Ended: 2008/07/02 13:11:54.755371
```

```
*****
```

```
MVS TCP/IP NETSTAT CS V1R9      TCPIP Name: TCPIP      13:11:54
```

```
Tcpip started at 01:28:36 on 06/23/2008 with IPv6 enabled
```

```
-----
host IP address:
-----
```

```
hostName=CDFMVS08
hostAddr=9.42.112.75
bindAddr=9.42.112.75
localAddr=9.42.112.75
```

```
Success, addresses match
```

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 that in the following sample (\$ is the z/OS UNIX prompt):

```
$ fekfivpd 4035 USERID
```

```
Wed Jul  2 15:00:27 EDT 2008  
uid=1(USERID) gid=0(GROUP)  
using /etc/rdz/rsed.envvars
```

```
current address space size limit is 1914675200 (1826.0 MB)  
maximum address space size limit is 2147483647 (2048.0 MB)
```

```
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 that in the following sample (\$ is the z/OS UNIX prompt):

```
$ fekfivpj 6715
```

```
Wed Jul  2 15:00:27 EDT 2008  
uid=1(USERID) gid=0(GROUP)  
using /etc/rdz/rsed.envvars
```

```
current address space size limit is 1914675200 (1826.0 MB)  
maximum address space size limit is 2147483647 (2048.0 MB)
```

```
hostName=CDFMVS08  
hostAddr=9.42.112.75  
Waiting for JES Job Monitor response...  
ACKNOWLEDGE01v03
```

```
Success
```

Lock daemon connection

Verify the lock daemon connection by executing the following command.

```
fekfivpl
```

The command should return an output like that in the following sample (\$ is the z/OS UNIX prompt):

```
$ fekfivpl
```

```
Mon Jun 29 08:00:38 EDT 2009  
uid=1(USERID) gid=0(GROUP)  
using /etc/rdz/rsed.envvars
```

```
current address space size limit is 1914675200 (1826.0 MB)  
maximum address space size limit is 2147483647 (2048.0 MB)
```

```
hostName=CDFMVS08  
hostAddr=9.42.112.75
```

```
Registering user to Lock Daemon...
Waiting for Lock Daemon response...
```

```
Querying to Lock Daemon...
Waiting for Lock Daemon response...
USERID
```

```
Unregistering user from Lock Daemon...
Waiting for Lock Daemon response...
```

```
Querying to Lock Daemon...
Waiting for Lock Daemon response...
```

Success

ISPF's TSO/ISPF Client Gateway connection

Verify the connection to ISPF's TSO/ISPF client Gateway by executing the following command:

```
fekfivpi
```

The command should return the result of ISPF's TSO/ISPF client Gateway-related checks (variables, HFS modules, starting and stopping TSO/ISPF session), like that in the following sample (\$ is the z/OS UNIX prompt):

```
$ fekfivpi
```

```
Wed Jul  2 15:00:27 EDT 2008
uid=1(USERID) gid=0(GROUP)
using /etc/rdz/rsed.envvars
```

```
current address space size limit is 1914675200 (1826.0 MB)
maximum address space size limit is 2147483647 (2048.0 MB)
```

```
-----
/etc/rdz/ISPF.conf content:
-----
```

```
isplib=ISP.SISPMENU
isptlib=ISP.SISPTENU
ispplib=ISP.SISPPENU
ispslib=ISP.SISPSLIB
sysproc=ISP.SISPCLIB,FEK.SFEKPROC
```

```
-----
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/rdz/lib:/usr/lpp/ispf/bin:
/bin:/usr/sbin:.
```

```
STEPLIB              = FEK.SFEKAUTH:FEK.SFEKLOAD
```

```
_CMDSERV_BASE_HOME   = /usr/lpp/ispf
_CMDSERV_CONF_HOME    = /etc/rdz
_CMDSERV_WORK_HOME    = /var/rdz
```

```
-----
*** CHECK : USS MODULES
```

```

Checking ISPF Directory : /usr/lpp/ispf
Checking modules in /usr/lpp/ispf/bin directory
Checking for ISPF configuration file ISPF.conf
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 ISPF checks fail, more detailed information will be shown.

fekfivpi has the following optional, non-positional, parameters:

-file fekfivpi can produce large amounts of output (hundreds of lines). The -file parameter sends this output to a file, userlog/.eclipse/RSE/\$LOGNAME/fekfivpi.log, where userlog is the value of the user.log directive in rsed.envvars, and \$LOGNAME is your user ID (uppercase). If the user.log directive is commented out or not present, your home path is used. The home path is defined in your OMVS security segment.

-debug

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

(Optional) TSO Commands service connection using APPC

Verify the connection to the TSO Commands 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 that in the following sample (\$ is the z/OS UNIX prompt):

```
$ fekfivpa USERID
Enter password:
```

```

Wed Jul  2 15:00:27 EDT 2008
uid=1(USERID) gid=0(GROUP)
using /etc/rdz/rsed.envvars

```

```

current address space size limit is 1914675200 (1826.0 MB)
maximum address space size limit is 2147483647 (2048.0 MB)

```

```

20070607 13:57:18.584060 /usr/lpp/rdz/bin/fekfscmd: version=Oct 2003
20070607 13:57:18.584326 Input parms: 1.2.3.4 * NOTRACE USERID *****
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

```

(Optional) SCLMDT connection

Verify the connection to SCLM Developer Toolkit by executing the following command:

```
fekfivps
```

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

```
$ fekfivps
```

```

Wed Jul  2 15:00:27 EDT 2008
uid=1(USERID) gid=0(GROUP)
using /etc/rdz/rsed.envvars

```

```

current address space size limit is 1914675200 (1826.0 MB)
maximum address space size limit is 2147483647 (2048.0 MB)

```

```
-----
/etc/rdz/ISPF.conf content:
-----
```

```

isplib=ISP.SISPMENU
isptlib=ISP.SISPTENU
ispplib=ISP.SISPPENU
ispslib=ISP.SISPSLIB
sysproc=ISP.SISPCLIB,FEK.SFEKPROC
-----

```

```

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

```

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

```

Server PATH          = /usr/lpp/java/J5.0/bin:/usr/lpp/rdz/lib:/usr/lpp/ispf/bin:
/bin:/usr/sbin:.

```

```
STEPLIB              = FEK.SFEKAUTH:FEK.SFEKLOAD
```

```

_CMDSERV_BASE_HOME  = /usr/lpp/ispf
_CMDSERV_CONF_HOME  = /etc/rdz
_CMDSERV_WORK_HOME   = /var/rdz
_SCLMDT_CONF_HOME    = /var/rdz/sclmdt
_SCLMDT_WORK_HOME    = /var/rdz
_SCLMDT_TRANTABLE    = FEK.#CUST.LSTRANS.FILE

```

```

-----
*** CHECK : JAVA PATH SETUP VERIFICATION
RC=0
MSG: SUCCESSFUL
-----

```

```
*** CHECK : USS MODULES
```

```

Checking ISPF Directory : /usr/lpp/ispf
Checking modules in /usr/lpp/ispf/bin directory
Checking for ISPF configuration file ISPF.conf
Checking install bin Directory : /usr/lpp/rdz/bin
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.

fekfivps has the following optional, non-positional, parameters:

-file fekfivps can produce large amounts of output (hundreds of lines). The -file parameter sends this output to a file, userlog/.eclipse/RSE/\$LOGNAME/fekfivps.log, where userlog is the value of the user.log directive in rsed.envvars, and \$LOGNAME is your user ID (uppercase). If the user.log directive is commented out or not present, your home path is used. The home path is defined in your OMVS security segment.

-debug

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

(Optional) 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 that in the following sample (\$ is the z/OS UNIX prompt):

```
$ fekfivpr 512 USERID
Enter password:
```

```

Wed Jul  2 15:00:27 EDT 2008
uid=1(USERID) gid=0(GROUP)
using /etc/rdz/rsed.envvars

```

```

current address space size limit is 1914675200 (1826.0 MB)
maximum address space size limit is 2147483647 (2048.0 MB)

```

```

$ EZYRC01I Calling function rexec_af with the following:
EZYRC02I Host: CDFMVS08, user USERID, cmd cd /etc/rdz;export RSE_USER_ID=USERI

```

```
D;./server.zseries -ivp, port 512
EZYRC19I Data socket = 4, Control socket = 6.
```

RSE server IVP test

```
CDFMVS08 -- Wed Jul 2 15:00:27 EDT 2008
uid=1(USERID) gid=0(GROUP)
```

RSE configuration files located in /etc/rdz --default

RSE userid is USERID --default

```
-----
Address Space size limits
```

```
-----
current address space size limit is 2147483647 (2048.0 MB)
maximum address space size limit is 2147483647 (2048.0 MB)
```

```
-----
service history
```

```
-----
Fri Jun 19 00:01:00 2009 -- COPY -- HHOP760 v7600 created 18 Jun 2009
```

```
-----
expect to see time out messages after a successful IVP test
```

```
-----
starting RSE server in background -- Fri Jun 19 15:59:05 EDT 2009
-----
```

```
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 (JI
T enabled)
J9VM - 20070131_11312_bHdSMr
JIT - 20070109_1805ifx1_r8
GC - 200701_09)
JCL - 20070126
```

```
DStore Server Starting...
Server Started Successfully
8108
Server running on: CDFMVS08
```

Note:

- If you do not 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).
- You can test the shell script used by REXEC separately, as described in the next IVP test, “(Optional) REXEC/SSH shell script.”
- The server is started without a client trying to connect, so it will time out (after 5 seconds). This results in a Connection error message that looks like the following sample:

```
Connection error
Server: error initializing socket: java.net.SocketTimeoutException:
Accept timed out
```

(Optional) REXEC/SSH shell script

This IVP test can be skipped if the previous test outlined in, “(Optional) REXEC connection” on page 109, completed successfully.

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

fekfivpz

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

```
$ fekfivpz
```

```
Wed Jul  2 15:00:27 EDT 2008
uid=1(USERID) gid=0(GROUP)
```

```
using /etc/rdz/rsed.envvars
```

```
current address space size limit is 1914675200 (1826.0 MB)
maximum address space size limit is 2147483647 (2048.0 MB)
```

```
-----
RSE server IVP test
```

```
CDFMVS08 -- Wed Jul  2 15:00:27 EDT 2008
uid=1(USERID) gid=0(GROUP)
```

```
RSE configuration files located in /etc/rdz --default
RSE userid is USERID --default
```

```
-----
Address Space size limits
```

```
-----
current address space size limit is 2147483647 (2048.0 MB)
maximum address space size limit is 2147483647 (2048.0 MB)
```

```
-----
service history
```

```
-----
Fri Jun 19 00:01:00 2009 -- COPY -- HHOP760 v7600 created 18 Jun 2009
```

```
-----
expect to see time out messages after a successful IVP test
```

```
-----
starting RSE server in background -- Fri Jun 19 15:59:05 EDT 2009
```

```
-----
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 j9vmz3123-20070201 (JIT enabled))
J9VM - 20070131_11312_bHdSMr
JIT  - 20070109_1805ifx1_r8
GC   - 200701_09)
JCL  - 20070126
```

```
DStore Server Starting...
Server Started Successfully
8108
Server running on: CDFMVS08
```

Note:

- If you do not get any output, your (TSO) region size is probably too small (must be 2096128).
- The server is started without a client trying to connect, so it will time out (after 5 seconds). This results in a Connection error message that looks like the following sample:

```
Connection error
Server: error initializing socket: java.net.SocketTimeoutException:
Accept timed out
```

Part 2. Developer for System z information

Chapter 8. Operator commands

This chapter provides an overview of the available operator (or console) commands for Developer for System z. Refer to “How to read a syntax diagram” on page 123 if you are unfamiliar with the syntax diagrams used to explain the command format.

Start (S)

Use the **START** command to dynamically start a started task (STC). The abbreviated version of the command is the letter S.

JES Job Monitor

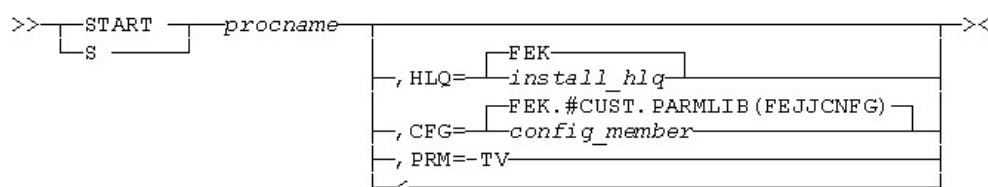


Figure 33. START JMON operator command

procname

The name of the member in a procedure library that is used to start the server. The default name used during the host configuration is JMON.

HLQ=install_hlq

High-level qualifier used to install Developer for System z. The default is FEK.

CFG=config_member

Absolute data set and member name of the JES Job Monitor configuration file. The default is FEK.#CUST.PARMLIB(FEJJCNFG).

PRM=-TV

Enable verbose (trace) mode. Tracing will cause performance degradations and should only be done under the direction of the IBM support center.

RSE daemon

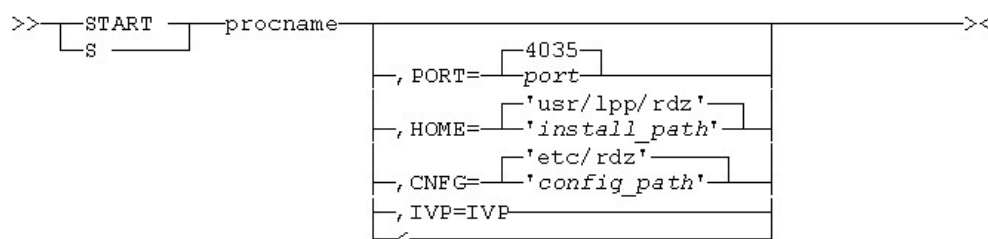


Figure 34. START RSED operator command

procname

The name of the member in a procedure library that is used to start the server. The default name used during the host configuration is RSED.

PORT=port

The port used by the RSE daemon for the clients to connect. The default is 4035.

HOME='install_path'

Path prefix and the mandatory /usr/lpp/rdz used to install Developer for System z. The default is '/usr/lpp/rdz'. Note that the z/OS UNIX path is case sensitive and that it must be enclosed in single quotes (') to preserve lower case characters.

CNFG='config_path'

Absolute location of the configuration files stored in z/OS UNIX. The default is '/etc/rdz'. Note that the z/OS UNIX path is case sensitive and that it must be enclosed in single quotes (') to preserve lower case characters.

IVP=IVP

Do not start the server but run the RSE daemon installation verification program (IVP).

Lock daemon

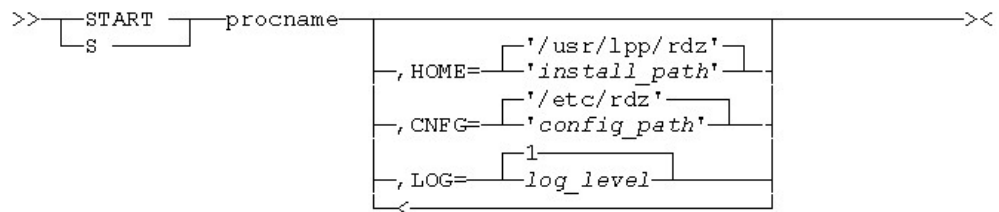


Figure 35. START LOCKD operator command

procname

The name of the member in a procedure library that is used to start the server. The default name used during the host configuration is LOCKD.

HOME='install_path'

Path prefix and the mandatory /usr/lpp/rdz used to install Developer for System z. The default is '/usr/lpp/rdz'. Note that the z/OS UNIX path is case sensitive and that it must be enclosed in single quotes (') to preserve lower case characters.

CNFG='config_path'

Absolute location of the configuration files stored in z/OS UNIX. The default is '/etc/rdz'. Note that the z/OS UNIX path is case sensitive and that it must be enclosed in single quotes (') to preserve lower case characters.

LOG=log_level

The detail level of output in DD STDOUT.

- 0 : Log error messages only.
- 1 : Log error and warning messages (default).
- 2 : Log error, warning and informational messages.

Modify (F)

The **MODIFY** command allows you to dynamically query and change the characteristics of an active task. The abbreviated version of the command is the letter F.

JES Job Monitor

```
>> MODIFY procname , APPL=-TV  
      F      , APPL=-TN <<
```

Figure 36. *MODIFY JMON operator command*

procname

The name of the member in a procedure library that was used to start the server. The default name used during the host configuration is JMON.

- TV** Enable verbose (trace) mode. Tracing will cause performance degradations and should only be done under the direction of the IBM support center.
- TN** Disable verbose (trace) mode.

RSE daemon

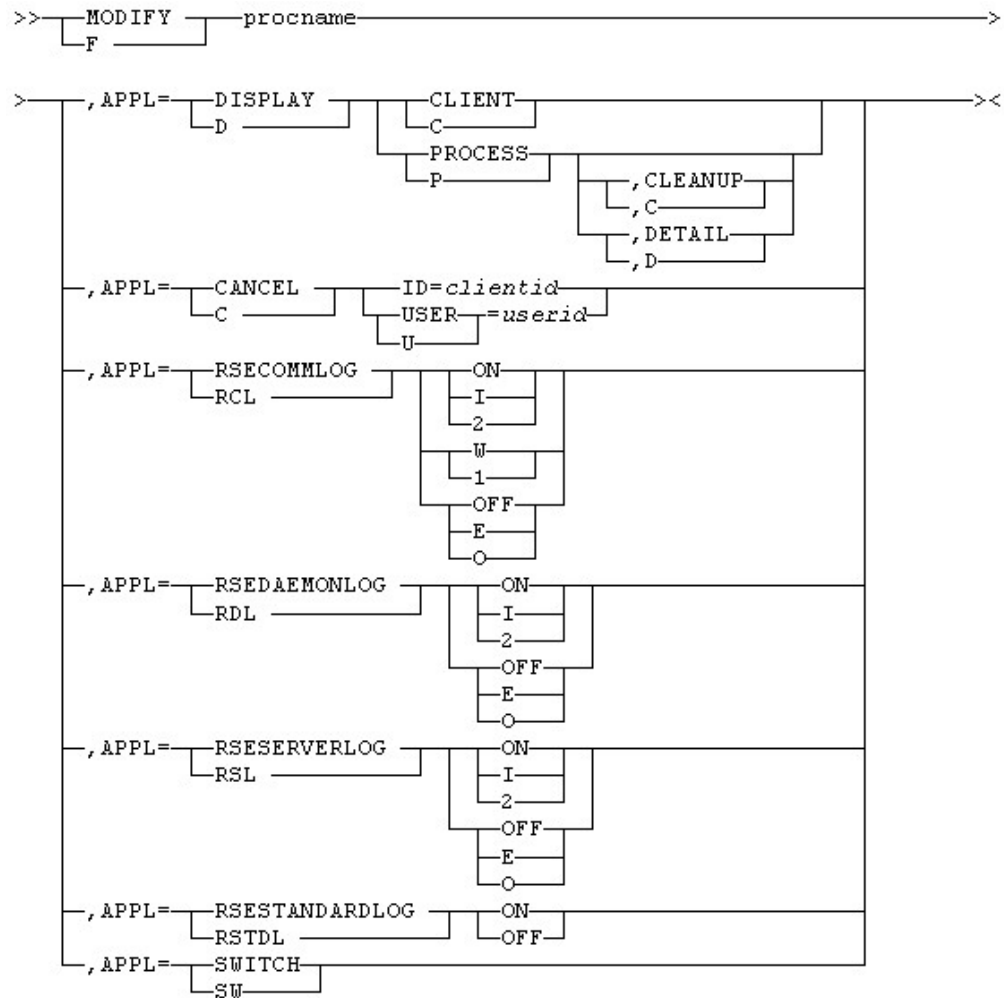


Figure 37. MODIFY RSED operator command

procname

The name of the member in a procedure library that was used to start the server. The default name used during the host configuration is RSED.

DISPLAY CLIENT

Display the active clients.

<clientid> : <userid> : <connected since>

DISPLAY PROCESS[,CLEANUP,DETAIL]

Display the RSE thread pool processes. There can be multiple processes, which are used for load balancing the connected users.

ProcessId(<processid>) Memory Usage(<java heap usage>%)
Clients(<number of clients>) Order(<startup order>) <error status>

Note:

- <processid> can be used in process specific z/OS UNIX operator commands.
- Each process has its own Java heap, whose size can be set in rsed.envvars.

- <startup order> is a sequential number that indicates the order that the thread pools were started. The number corresponds to the number used in the filename of the stderr.*.log and stdout.*.log files.

In normal situations, <error status> is blank. Table 18 documents the possible non-blank values for <error status>.

Table 18. Thread pool error status

Status	Description
severe error	The thread pool process encountered an unrecoverable error and halted operations. The other status fields show the last known values. Use the CLEANUP option of the DISPLAY PROCESS modify command to remove this entry from the table.
killed process	The thread pool process was killed by Java, z/OS UNIX or an operator command. The other status fields show the last known values. Use the CLEANUP option of the DISPLAY PROCESS modify command to remove this entry from the table.
timeout	The thread pool process did not respond in a timely manner to RSE daemon during a client connect request. The other status fields show the current values. The thread pool is excluded for future client connect requests. The *timeout* status is reset when a client served by this thread pool logs off.

More information is provided when the DETAIL option of the **DISPLAY PROCESS** modify command is used:

```

ProcessId(33555087) ASId(002E) JobName(RSED8) Order(1)
PROCESS LIMITS:  CURRENT  HIGHWATER  LIMIT
JAVA HEAP USAGE(%)    10         56       100
CLIENTS                0         25        60
MAXFILEPROC            83        103     64000
MAXPROCUSER            97         99       200
MAXTHREADS             9         14      1500
MAXTHREADTASKS         9         14      1500

```

The ASId field is the address space ID, in hexadecimal notation. The process limits table shows the current resource usage, the high-water mark for the resource usage, and the resource limit. Note that due to other limiting factors, the defined limit might never be reached.

CANCEL ID=clientid

Cancel a client connection based upon the client ID, which is shown in the **DISPLAY CLIENT** modify command.

CANCEL USER=userid

Cancel a client connection based upon the client's user ID, which is shown in the **DISPLAY CLIENT** modify command.

RSECOMMLOG {ON,OFF,I,W,E,2,1,0}

Control the trace detail level for RSE server (rsecomm.log) and the MVS

data set services (lock.log and ffs*.log). The startup default is defined in rsecomm.properties. There are three detail levels available:

E or 0 or OFF	Error messages only.
W or 1	Error and Warning messages. This is the default setting in rsecomm.properties.
I or 2 or ON	Error, Warning and Informational messages.

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

RSEDAEMONLOG {ON,OFF,I,E,2,0}

Control the trace detail level for RSE daemon (rsedaemon.log). The startup default is defined in rsecomm.properties. There are two detail levels available:

E or 0 or OFF	Error messages only.
I or 2 or ON	Error, Warning, and Informational messages.

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

RSESERVERLOG {ON,OFF,I,E,2,0}

Control the trace detail level for RSE thread pools (rseserver.log). The startup default is defined in rsecomm.properties. There are two detail levels available:

E or 0 or OFF	Error messages only.
I or 2 or ON	Error, Warning, and Informational messages.

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

RSESTANDARDLOG {ON,OFF}

Disable (OFF) or enable (ON) updating the log files holding the stdout and stderr streams of the thread pools (stdout*.log and stderr*.log). The startup default is defined by the enable.standard.log directive in rsed.envvars.

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

SWITCH

Switch to a new audit log file.

Note:

- Refer to “Log files” on page 128 in Chapter 9, “Troubleshooting configuration problems,” on page 127 for more information on the log files mentioned above.
- Refer to “Audit logging” on page 152 in Chapter 10, “Security considerations,” on page 147 for more information on audit.

Lock daemon

```
>> MODIFY procname >
  F
> , APPL= QUERY dataset ><
  Q dataset (member)
```

Figure 38. MODIFY LOCKD operator command

procname

The name of the member in a procedure library that was used to start the server. The default name used during the host configuration is LOCKD.

QUERY dataset[(member)]

Query the lock status of the listed data set or member. The server will reply with one of the following messages:

```
BPXM023I (stclock) dataset[(member)] NOT LOCKED
BPXM023I (stclock) dataset[(member)] LOCKED BY userid
```

Note:

- The server will also report locks held by other products, such as ISPF.
- Locks held by Developer for System z clients who were unable to register with the lock daemon will result in the thread pool server address space (RSEDx) being reported as lock owner.

Console message FEK513W is generated when RSE server is unable to register the client with the lock daemon. The ASID and TCB values mentioned in this message can be compared against the output of the **D GRS,RES=(*,dataset[(member))]** operator command in order to find the actual user holding the lock.

Stop (P)

Use the **STOP** command to stop an active task. The abbreviated version of the command is the letter P.

```
>> STOP procname ><
  P
```

Figure 39. STOP operator command

procname

The name of the member in a procedure library that was used to start the server. The default names used during the host configuration are JMON, RSED, and LOCKD for JES Job Monitor, RSE daemon, and the lock daemon, respectively.

Console messages

JES Job Monitor

JES Job Monitor does not have product-specific console messages. The server relies on z/OS and JES to generate console messages for actions done by Developer for System z clients.

RSE daemon, RSE thread pool server, and lock daemon

Table 19 lists the product-specific console messages generated by RSE daemon, RSE thread pool server, and the lock daemon.

Table 19. RSE console messages

Message ID	Message text
FEK001I	RseDaemon being initialized in {0} bit mode
FEK002I	RseDaemon started. (port={0})
FEK003I	Stop command being processed
FEK004I	RseDaemon: Max Heap Size={0}MB and private AS Size={1}MB
FEK005I	Server process started. (processId={0})
FEK009I	RseDaemon is waiting for the server process to start.
FEK010I	(rsed.envvars location = {0})
FEK011I	(log directory = {0})
FEK100E	Daemon port/timeout value must be digits
FEK101E	JRE {0} or higher required
FEK102E	Invalid arguments received: {0}
FEK103E	Almost Disk-Full in {0}
FEK104E	Maximum number of processes has been reached
FEK105E	Error in sending audit data (rc={0})
FEK110E	socket() failed. reason=({0})
FEK111E	setsockopt() failed. reason=({0})
FEK112E	bind() failed. reason=({0})
FEK113E	listen() failed. reason=({0})
FEK114E	accept() failed. reason=({0})
FEK115E	write() failed. reason=({0})
FEK116E	pipe() failed. reason=({0})
FEK117E	socketpair() failed. reason=({0})
FEK118E	select() failed. reason=({0})
FEK119E	_console() failed. reason=({0})
FEK130E	gsk_environment_open() failed. reason=({0})
FEK131E	gsk_attribute_set_enum(GSK_PROTOCOL_SSLV2) failed. reason=({0})
FEK132E	gsk_attribute_set_enum(GSK_PROTOCOL_SSLV3) failed. reason=({0})
FEK133E	gsk_attribute_set_enum(GSK_PROTOCOL_TLSV1) failed. reason=({0})
FEK134E	gsk_attribute_set_buffer(GSK_KEYRING_FILE) failed. reason=({0})
FEK135E	gsk_attribute_set_buffer(GSK_KEYRING_PW) failed. reason=({0})
FEK136E	gsk_environment_init() failed. reason=({0})
FEK137E	gsk_secure_socket_open() failed. reason=({0})
FEK138E	gsk_attribute_set_numeric_value(GSK_FD) failed. reason=({0})
FEK139E	gsk_attribute_set_buffer(GSK_KEYRING_LABEL) failed. reason=({0})
FEK140E	gsk_attribute_set_enum(GSK_SESSION_TYPE) failed. reason=({0})
FEK141E	gsk_attribute_set_callback(GSK_IO_CALLBACK) failed. reason=({0})
FEK142E	gsk_secure_socket_init() failed. reason=({0})

Table 19. RSE console messages (continued)

Message ID	Message text
FEK143E	gsk_attribute_set_enum(GSK_CLIENT_AUTH_TYPE) failed. reason=({0})
FEK144E	gsk_get_cert_info failed. reason=({0})
FEK145E	gsk_secure_socket_read() failed. reason=({0})
FEK146E	gsk_secure_socket_write() failed. reason=({0})
FEK150E	RseDaemon abnormally terminated; {0}
FEK201I	{0} Command has been processed
FEK202E	Invalid Command Entered
FEK203E	Invalid Display Command: Display Process Client
FEK204E	Invalid Cancel Command: Cancel ID= User=
FEK205E	Command was not processed owing to consecutive SWITCHs
FEK206E	Audit Log facility is not active
FEK207I	No Client to be displayed
FEK208I	{0} canceled
FEK209I	No Process to be displayed
FEK210I	{0} canceled owing to duplicate logon
FEK501I	Lock daemon started, port={0}, cleanup interval={1}, log level={2}
FEK502I	Lock daemon terminating
FEK510E	Lock daemon, missing port
FEK511E	Lock daemon, wrong port, port={0}
FEK512E	Lock daemon, socket error, port={0}
FEK513W	Lock daemon, registration failed, ASID={0}, TCB={1}, USER={2}
FEK514W	Lock daemon, wrong log level, log level={0}
BPXM023I	(stclock) dataset[(member)] NOT LOCKED
BPXM023I	(stclock) dataset[(member)] LOCKED BY userid
BPXM023I	(stclock) command, WRONG COMMAND
BPXM023I	(stclock) command, MISSING ARGUMENT
BPXM023I	(stclock) argument, WRONG ARGUMENT

How to read a syntax diagram

The syntax diagram shows you how to specify a command so that the operating system can correctly interpret what you type. Read the syntax diagram from left to right and from top to bottom, following the horizontal line (the main path).

Symbols


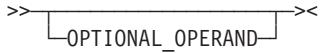
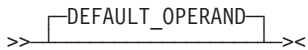
The following symbols are used in syntax diagrams:

Symbol	Description
>>	Marks the beginning of the syntax diagram.
>	Indicates that the syntax diagram is continued.

Symbol	Description
	Marks the beginning and end of a fragment or part of the syntax diagram.
><	Marks the end of the syntax diagram.

Operands

The following types of operands are used in syntax diagrams:

- Required operands are displayed on the main path line:

- Optional operands are displayed below the main path line:

- Default operands are displayed above the main path line:


Operands are classified as keywords or variables:

- Keywords are constants that must be provided. If the keyword appears in the syntax diagram in both uppercase and lowercase, the uppercase portion is the abbreviation for the keyword (for example, KEYword). Keywords are not case-sensitive. You can code them in uppercase or lowercase.
- Variables are italicized, appear in lowercase letters, and represent names or values you supply. For example, a data set name is a variable. Variables can be case sensitive.

Syntax example

In the following example, the USER command is a keyword. The required variable parameter is *user_id*, and the optional variable parameter is *password*. Replace the variable parameters with your own values:

```
>>—USER—user_id—┐—————><
                  └──┬──password──┘
```

Nonalphanumeric characters and blank spaces

If a diagram shows a character that is not alphanumeric (such as parentheses, periods, commas, equal signs, and blank spaces), you must code the character as part of the syntax. In this example, you must code OPERAND=(001 0.001):

```
>>—OPERAND—==(—001— —0.001—)—————><
```

Selecting more than one operand

An arrow returning to the left in a group of operands means that more than one can be selected, or that a single one can be repeated:

```
>>—┐—————><
    └──┬──REPEATABLE_OPERAND_1──┘
      └──┬──REPEATABLE_OPERAND_2──┘
        <—
```

Longer than one line

If a diagram is longer than one line, the first line ends with a single arrowhead and the second line begins with a single arrowhead:

>>—| The first line of a syntax diagram that is longer than one line |—>
>—| The continuation of the subcommands, parameters, or both |————><

Syntax fragments

Some diagrams might contain syntax fragments, which serve to break up diagrams that are too long, too complex, or too repetitious. Syntax fragment names are in mixed case and are shown in the diagram and in the heading of the fragment. The fragment is placed below the main diagram:

>>—| Syntax fragment |————><

Syntax fragment:

|—1ST_OPERAND—,—2ND_OPERAND—,—3RD_OPERAND—|

Chapter 9. Troubleshooting configuration problems

This chapter is provided to assist you with some common problems that you may encounter during your configuration of Developer for System z, and has the following sections:

- “Log and setup analysis using FEKLOGS”
- “Log files” on page 128
- “Dump files” on page 133
- “Tracing” on page 135
- “z/OS UNIX permission bits” on page 137
- “Reserved TCP/IP ports” on page 140
- “Address Space size” on page 141
- “APPC transaction and TSO Commands service” on page 142
- “Miscellaneous information” on page 144

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

In the Library section of the Web site (<http://www-306.ibm.com/software/awdtools/rdz/library/>) you can also find the latest version of the Developer for System z documentation, including whitepapers.

The Developer for System z Information Center (<http://publib.boulder.ibm.com/infocenter/ratdevz/v7r6/index.jsp>) documents the Developer for System z client, and how it interacts with the host (from a client's perspective).

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

Please notify us if you think that Developer for System z misses a certain function. You can open a Request For Enhancement (RFE) at <https://www.ibm.com/developerworks/support/rational/rfe/>

Log and setup analysis using FEKLOGS

Developer for System z provides a sample job, FEKLOGS, which gathers all z/OS UNIX log files as well as Developer for System z installation and configuration information.

Sample job FEKLOGS is located in FEK.#CUST.JCL, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details.

The customization of FEKLOGS is described within the JCL. The customization encompasses the provision of a few key variables.

Note: SDSF customers can use the XDC line command in SDSF to save the job output in a data set, which in turn can be given to the IBM support center.

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 on your z/OS host system. 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:

- userlog/\$LOGNAME/

User-specific log files are located in userlog/\$LOGNAME/, where userlog is the combined value of the user.log and DSTORE_LOG_DIRECTORY directives in rsed.envvars, and \$LOGNAME is the logon user ID (uppercase). If the user.log directive is commented out or not present, the home path of the user is used. The home path is defined in the OMVS security segment of the user ID. If the DSTORE_LOG_DIRECTORY directive is commented out or not present, then .eclipse/RSE/ is appended to the user.log value.

- .dstoreMemLogging - DataStore memory usage logging
- .dstoreTrace - DataStore action logging
- fa.log - The log of the Fault Analyzer Integration
- fekfivpi.log - The log of the fekfivpi IVP test
- fekfivps.log - The log of the fekfivps IVP test
- ffs.log - The log of the Foreign File System (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 and the communication logging of all services relying on RSE (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

Note: The .eclipse directory and the .dstore* log files start with a dot (.), which makes them hidden. Use z/OS UNIX command **ls -IA** to list hidden files and directories. When using the Developer for System z client, select the **Window > Preferences... > Remote Systems > Files** preference page and enable "Show hidden files".

- daemon-home

The RSE daemon and RSE thread pool specific log files are located in daemon-home, where daemon-home is the value of the daemon.log directive in rsed.envvars. If the daemon.log directive is commented out or not present, the home directory of the user ID assigned to the RSED started task is used. The home directory is defined in the OMVS security segment of the user ID.

- rsedaemon.log - The log of the RSE daemon
- rseserver.log - The log of the RSE thread pools

- audit.log - The RSE audit trail
- serverlogs.count - Counter for logging RSE thread pool streams
- stderr.*.log - RSE thread pool standard error stream
- stdout.*.log - RSE thread pool standard output stream

Note: There are operator commands available to control the amount of data written to some of the mentioned log files. Refer to Chapter 8, “Operator commands,” on page 115 for more information.

JES Job Monitor logging

- **SYSOUT DD**
Logging of normal operations. The default value in the sample JCL FEK.#CUST.PROCLIB(JMON) is SYSOUT=*.
- **SYSPRINT DD**
Trace logging. The default value in the sample JCL FEK.#CUST.PROCLIB(JMON) is SYSOUT=*. Tracing is activated with the -TV parameter, see “JES Job Monitor tracing” on page 135 for more details.

Lock daemon logging

- **STDOUT DD**
The redirected data of stdout, Java standard output. The default value in the sample JCL FEK.#CUST.PROCLIB(LOCKD) is SYSOUT=*.
- **STDERR DD**
The redirected data of stderr, Java standard error output. The default value in the sample JCL FEK.#CUST.PROCLIB(LOCKD) is SYSOUT=*.

RSE daemon and thread pool logging

- **STDOUT DD**
The redirected data of stdout, Java standard output of RSE daemon. The default value in the sample JCL FEK.#CUST.PROCLIB(RSED) is SYSOUT=*.
- **STDERR DD**
The redirected data of stderr, Java standard error output of RSE daemon. The default value in the sample JCL FEK.#CUST.PROCLIB(RSED) is SYSOUT=*.
- **daemon-home**
The RSE daemon and RSE thread pool specific log files are located in daemon-home, where daemon-home is the value of the daemon.log directive in rsed.envvars. If the daemon.log directive is commented out or not present, the home directory of the user ID assigned to the RSED started task is used. The home directory is defined in the OMVS security segment of the user ID.
 - rsedaemon.log - The log of the RSE daemon
 - rseserver.log - The log of the RSE thread pools
 - audit.log - The RSE audit trail
 - serverlogs.count - Counter for logging RSE thread pool streams
 - stderr.*.log - RSE thread pool standard error stream
 - stdout.*.log - RSE thread pool standard output stream

Note:

- `serverlogs.count`, `stderr.*.log`, and `stdout.*.log` are only created if the `enable.standard.log` directive in `rsed.envvars` is active, or if the function is dynamically activated with the **modify rsestandardlog on** operator command.
- The `*` in `stderr.*.log` and `stdout.*.log` is 1 by default. However, there can be multiple RSE thread pools, in which case the number is incremented for each new RSE thread pool to ensure unique file names.
- There are no user-specific `stdout.log` and `stderr.log` log files when the `enable.standard.log` directive is active. The user-specific data is now written to the matching RSE thread pool stream.
- There are operator commands available to control the amount of data written to some of the mentioned log files. Refer to Chapter 8, “Operator commands,” on page 115 for more information.

RSE user logging

- **userlog/\$LOGNAME/**

There are several log files created by the components related to RSE. All are located in `userlog/$LOGNAME/`, where `userlog` is the combined value of the `user.log` and `DSTORE_LOG_DIRECTORY` directives in `rsed.envvars`, and `$LOGNAME` is the logon user ID (uppercase). If the `user.log` directive is commented out or not present, the home path of the user is used. The home path is defined in the OMVS security segment of the user ID. If the `DSTORE_LOG_DIRECTORY` directive is commented out or not present, then `.eclipse/RSE/` is appended to the `user.log` value.

- `.dstoreMemLogging` - DataStore memory usage logging
- `.dstoreTrace` - DataStore action logging
- `ffs.log` - The log of the Foreign File System (FFS) server, which 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 or 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 and the communication logging of all services relying on RSE (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

Note:

- The `.eclipse` directory and the `.dstore*` log files start with a dot (`.`), which makes them hidden. Use z/OS UNIX command `ls -lA` to list hidden files and directories. When using the Developer for System z client, select the **Window > Preferences... > Remote Systems > Files** preference page and enable “Show hidden files”.
- The creation of the `.dstore*` log files is controlled by the `-DDSTORE_*` Java startup options, as described in “Defining extra Java startup parameters with `_RSE_JAVAOPTS`” on page 37.

- The .dstore* log files are created in ASCII. Use z/OS UNIX command **iconv -f ISO8859-1 -t IBM-1047 .dstore*** to display them in EBCDIC (when using code page IBM-1047).
- There are no user-specific stdout.log and stderr.log log files when the enable.standard.log directive is active. The user-specific data is now written to the matching RSE thread pool stream.
- There are operator commands available to control the amount of data written to some of the mentioned log files. Refer to Chapter 8, “Operator commands,” on page 115 for more information.

Fault Analyzer Integration logging

- **userlog/\$LOGNAME/**

Fault Analyzer Integration logging, where userlog is the combined value of the user.log and DSTORE_LOG_DIRECTORY directives in rsed.envvars, and \$LOGNAME is the logon user ID (uppercase). If the user.log directive is commented out or not present, the home path of the user is used. The home path is defined in the OMVS security segment of the user ID. If the DSTORE_LOG_DIRECTORY directive is commented out or not present, then .eclipse/RSE/ is appended to the user.log value.

- fa.log - The log of the Fault Analyzer Integration
- rsecomm.log - Communication logging of Fault Analyzer Integration

File Manager Integration logging

- **userlog/\$LOGNAME/rsecomm.log**

Communication logging of File Manager Integration, where userlog is the combined value of the user.log and DSTORE_LOG_DIRECTORY directives in rsed.envvars, and \$LOGNAME is the logon user ID (uppercase). If the user.log directive is commented out or not present, the home path of the user is used. The home path is defined in the OMVS security segment of the user ID. If the DSTORE_LOG_DIRECTORY directive is commented out or not present, then .eclipse/RSE/ is appended to the user.log value.

SCLM Developer Toolkit logging

- **userlog/\$LOGNAME/rsecomm.log**

Communication logging of SCLM Developer Toolkit, where userlog is the combined value of the user.log and DSTORE_LOG_DIRECTORY directives in rsed.envvars, and \$LOGNAME is the logon user ID (uppercase). If the user.log directive is commented out or not present, the home path of the user is used. The home path is defined in the OMVS security segment of the user ID. If the DSTORE_LOG_DIRECTORY directive is commented out or not present, then .eclipse/RSE/ is appended to the user.log value.

CARMA logging

- **CARMA server job**

When opening a connection with CARMA, using the batch interface, FEK.#CUST.SYSPROC(CRASUBMT) 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 the chosen CARMA startup method, CARMA logging is redirected to this DD statement in the server job, otherwise it goes to SYSPRINT.

- **SYSPRINT DD**

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

- **userlog/\$LOGNAME/rsecomm.log**

Communication logging of CARMA, where userlog is the combined value of the user.log and DSTORE_LOG_DIRECTORY directives in rsed.envvars, and \$LOGNAME is the logon user ID (uppercase). If the user.log directive is commented out or not present, the home path of the user is used. The home path is defined in the OMVS security segment of the user ID. If the DSTORE_LOG_DIRECTORY directive is commented out or not present, then .eclipse/RSE/ is appended to the user.log value.

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 ATBPDFMU utility. The default value in sample JCL FEK.SFEKSAMP(FEKAAPPCC) is SYSOUT=*. Refer to *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 FEK.SFEKSAMP(FEKAAPPCC) is &SYSUID.FEKFRSRV.&TPDATE.&TPTIME.LOG. Refer to *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.

fekfivpi IVP test logging

- **userlog/\$LOGNAME/fekfivpi.log**

Output of the fekfivpi -file command (TSO/ISPF Client Gateway related IVP test), where userlog is the combined value of the user.log and DSTORE_LOG_DIRECTORY directives in rsed.envvars, and \$LOGNAME is the logon user ID (uppercase). If the user.log directive is commented out or not present, the home path of the user is used. The home path is defined in the OMVS security segment of the user ID. If the DSTORE_LOG_DIRECTORY directive is commented out or not present, then .eclipse/RSE/ is appended to the user.log value.

fekfivps IVP test logging

- **userlog/\$LOGNAME/fekfivps.log**

Output of the fekfivps -file command (SCLMDT-related IVP test), where userlog is the combined value of the user.log and DSTORE_LOG_DIRECTORY directives in rsed.envvars, and \$LOGNAME is the logon user ID (uppercase). If the user.log directive is commented out or not present, the home path of the user is used. The home path is defined in the OMVS security segment of the user ID. If the DSTORE_LOG_DIRECTORY directive is commented out or not present, then .eclipse/RSE/ is appended to the user.log value.

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

Java dumps

In z/OS UNIX, most 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 the following:

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 `%uid.JVM.TDUMP.%job.D%ym%d.T%H%M%S`, 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`.

Note: `JAVA_DUMP_TDUMP_PATTERN` allows the usage of variables, which are translated to an actual value at the time the transaction dump is taken.

Table 20. `JAVA_DUMP_TDUMP_PATTERN` variables

Variable	Usage
<code>%uid</code>	User ID
<code>%job</code>	Job name

Table 20. JAVA_DUMP_TDUMP_PATTERN variables (continued)

Variable	Usage
%y	Year (2 digits)
%m	Month (2 digits)
%d	Day (2 digits)
%H	Hour (2 digits)
%M	Minute (2 digits)
%S	Second (2 digits)

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 yyyymmdd 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 yyyymmdd 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, such as 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 yyyymmdd 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 *Java Diagnostic Guide* (SC34-6358) for more information on JVM dumps, and *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 /tmp. It can be changed to /dev/null to avoid creating the dump files.
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.

Tracing

JES Job Monitor tracing

JES Job Monitor tracing is controlled by the system operator, as described in Chapter 8, “Operator commands,” on page 115.

- Starting the JMON started task with the `PRM=-TV` parameter activates verbose mode (tracing)
- The **modify -TV** and **modify -TN** commands activate and deactivate tracing

RSE tracing

There are several log files created by the components related to RSE. Most are located in `userlog/$LOGNAME/`, where `userlog` is the combined value of the `user.log` and `DSTORE_LOG_DIRECTORY` directives in `rsed.envvars`, and `$LOGNAME` is the logon user ID (uppercase). If the `user.log` directive is commented out or not present, the home path of the user is used. The home path is defined in the OMVS security segment of the user ID. If the `DSTORE_LOG_DIRECTORY` directive is commented out or not present, then `.eclipse/RSE/` is appended to the `user.log` value.

The amount of data written to `ffs*.log`, `lock.log` and `rsecomm.log` is controlled by the **modify rsecommlog** operator command, or by setting `debug_level` in `rsecomm.properties`. See Chapter 8, “Operator commands,” on page 115 and “(Optional) RSE tracing” on page 88 for more details.

The creation of the `.dstore*` log files is controlled by the `-DDSTORE_*` Java startup options, as described in “Defining extra Java startup parameters with `_RSE_JAVAOPTS`” on page 37.

Note:

- The `.eclipse` directory and the `.dstore*` log files start with a dot (`.`), which makes them hidden. Use z/OS UNIX command `ls -lA` to list hidden files and directories. When using the Developer for System z client, select the **Window > Preferences... > Remote Systems > Files** preference page and enable “Show hidden files”.
- The `.dstore*` log files are created in ASCII. Use z/OS UNIX command `iconv -f ISO8859-1 -t IBM-1047 .dstore*` to display them in EBCDIC (when using code page IBM-1047).

The RSE daemon and RSE thread pool specific log files are located in `daemon-home`, where `daemon-home` is the value of the `daemon.log` directive in `rsed.envvars`. If the `daemon.log` directive is commented out or not present, the home directory of the user ID assigned to the RSED started task is used. The home directory is defined in the OMVS security segment of the user ID.

The amount of data written to `rsedaemon.log` and `rseserver.log` is controlled by the **modify rsedaemonlog** and **modify rseserverlog** operator commands or by setting `debug_level` in `rsecomm.properties`. See Chapter 8, “Operator commands,” on page 115 and “(Optional) RSE tracing” on page 88 for more details.

serverlogs.count, stderr.*.log, and stdout.*.log are only created if the enable.standard.log directive in rsed.envvars is active, or if the function is dynamically activated with the **modify rsestandardlog on** operator command..

Lock daemon tracing

The lock daemon-specific log is located in the STDOUT DD of the LOCKD started task. The amount of data written to the log is controlled by the LOG startup parameter. See Chapter 8, “Operator commands,” on page 115 and “(Optional) RSE tracing” on page 88 for more details.

CARMA tracing

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 the following:

Error Logging

Refer to “Log files” on page 128 for more information on log file locations.

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, such as SYS1.PROCLIB, as described in “ELAXF* remote build procedures” on page 22.
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 the COBOL program for which you want detailed tracing.
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.SF.Z682746.XML
23 //COBOL.WSEDSF2 DD DISP=MOD,
    // DSN=uid.ERRCOB.member.SF.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.

z/OS UNIX permission bits

Developer for System z requires that the z/OS UNIX file system and some z/OS UNIX files have certain permission bits set.

SETUID file system attribute

Remote Systems Explorer (RSE) is the Developer for System z component that provides core services such as connecting the client to the host. It must be allowed to perform tasks such as creating the user's security environment.

The file system (HFS or zFS) in which Developer for System z is installed must be mounted with the SETUID permission bit on (this is the system default). Mounting the file system with the NOSETUID parameter will prevent Developer for System z from creating the user's security environment, and will fail the connection request.

Use the TSO **ISHELL** command to list the current status of the SETUID bit. In the ISHELL panel, select **File_systems > 1. Mount table...** to list the mounted file systems. The **a** line command will show the attributes for the selected file system, where the "Ignore SETUID" field should be 0.

Program Control authorization

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

The z/OS UNIX program control bit is set during SMP/E install where needed, except for the Java interface to your security product, as documented in

Chapter 10, “Security considerations,” on page 147. This permission bit might get lost if you did not preserve it during a manual copy of the Developer for System z directories.

The following Developer for System z files must be program controlled:

- /usr/lpp/rdz/bin/
 - fekfdivp
 - fekfomvs
 - fekfrivp
- /usr/lpp/rdz/lib/
 - fekfdir.dll
 - libfekdcore.so
 - libfekfmain.so
- /usr/lpp/rdz/lib/icuc/
 - libicudata.dll
 - libicudata40.1.dll
 - libicudata40.dll
 - libicudata64.40.1.dll
 - libicudata64.40.dll
 - libicudata64.dll
 - libicuuc.dll
 - libicuuc40.1.dll
 - libicuuc40.dll
 - libicuuc64.40.1.dll
 - libicuuc64.40.dll
 - libicuuc64.dll

Note: The libicu*64.* files are only present if you applied the Developer for System z PTF that addresses APAR AM07305 to enable 64-bit support.

Use z/OS UNIX command **ls -E** to list the extended attributes, in which the program control bit is marked with the letter p, as shown in the following sample (\$ is the z/OS UNIX prompt):

```
$ cd /usr/lpp/rdz
$ ls -E lib/fekf*
-rwxr-xr-x -ps- 2 user      group      94208 Jul  8 12:31 lib/fekfdir.dll
```

Use z/OS UNIX command **extattr +p** to set the program control bit manually, as shown in the following sample (\$ and # are the z/OS UNIX prompt):

```
$ cd /usr/lpp/rdz
$ su
# extattr +p lib/fekf*
# exit
$ ls -E lib/fekf*
-rwxr-xr-x -ps- 2 user      group      94208 Jul  8 12:31 lib/fekfdir.dll
```

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, or be a superuser (UID 0) if this profile is not defined. For more information, refer to *UNIX System Services Planning* (GA22-7800).

APF authorization

Remote Systems Explorer (RSE) is the Developer for System z component that provides core services such as connecting the client to the host. It must run APF authorized in order to perform tasks such as displaying detailed process resource usage.

The z/OS UNIX APF bit is set during SMP/E install where needed. This permission bit might get lost if you did not preserve it during a manual copy of the Developer for System z directories.

The following Developer for System z files must be APF authorized:

- /usr/lpp/rdz/bin/
 - fekfomvs
 - fekfrivp

Use z/OS UNIX command **ls -E** to list the extended attributes, in which the APF bit is marked with the letter a, as shown in the following sample (\$ is the z/OS UNIX prompt):

```
$ cd /usr/lpp/rdz
$ ls -E bin/fekfrivp
-rwxr-xr-x  aps-  2 user      group      114688 Sep 17 06:41 bin/fekfrivp
```

Use z/OS UNIX command **extattr +a** to set the APF bit manually, as shown in the following sample (\$ and # are the z/OS UNIX prompts):

```
$ cd /usr/lpp/rdz
$ su
# extattr +a bin/fekfrivp
# exit
$ ls -E bin/fekfrivp
-rwxr-xr-x  aps-  2 user      group      114688 Sep 17 06:41 bin/fekfrivp
```

Note: To be able to use the **extattr +a** command, you must have at least READ access to the BPX.FILEATTR.APF profile in the FACILITY class of your security software, or be a superuser (UID 0) if this profile is not defined. For more information, refer to *UNIX System Services Planning* (GA22-7800).

Sticky bit

Some of the optional Developer for System z services require that MVS load modules are available to z/OS UNIX. This is done by creating a stub (a dummy file) in z/OS UNIX with the "sticky" bit on. When the stub is executed, z/OS UNIX will look for an MVS load module with the same name and execute the load module instead.

The z/OS UNIX sticky bit is set during SMP/E install where needed. These permission bits might get lost if you did not preserve them during a manual copy of the Developer for System z directories.

The following Developer for System z files must have the sticky bit on:

- /usr/lpp/rdz/bin/
 - BWBTSOW
 - CRASTART

Use z/OS UNIX command **ls -l** to list the permissions, in which the sticky bit is marked with the letter **t**, as shown in the following sample (\$ is the z/OS UNIX prompt):

```
$ cd /usr/lpp/rdz
$ ls -l bin/CRA*
-rwxr-xr-t  2 user      group          71 Jul  8 12:31 bin/CRASTART
```

Use z/OS UNIX command **chmod +t** to set the sticky bit manually, as shown in the following sample (\$ and # are the z/OS UNIX prompt):

```
$ cd /usr/lpp/rdz
$ su
# chmod +t bin/CRA*
# exit
$ ls -l bin/CRA*
-rwxr-xr-t  2 user      group          71 Jul  8 12:31 bin/CRASTART
```

Note: To be able to use the **chmod** command, you must have at least READ access to the SUPERUSER.FILESYS.CHANGEPERMS profile in the UNIXPRIV class of your security software, or be a superuser (UID 0) if this profile is not defined. For more information, refer to *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 the example below. The ports used are the last number (behind the ".") in the "Local Socket" column. Since these ports are already in use, they cannot be used for the Developer for System z configuration.

IPv4

MVS TCP/IP	NETSTAT	CS	VxRy	TCPIP Name:	TCPIP	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
RSED	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	VxRy	TCPIP Name:	TCPIP	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				
RSED	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 the following two 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 (multiple TCP/IP stacks active on a single host).
- 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 (multiple TCP/IP stacks active on a single host).

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 that in the example as follows:

MVS TCP/IP NETSTAT CS VxRy				TCP/IP Name: TCP/IP		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.

Address Space size

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

startup JCL requirements

The RSE daemon is started by JCL using BPXBATSL, whose region size must be 0.

Limitations set in SYS1.PARMLIB(BPXPRMxx)

Set MAXASSIZE in SYS1.PARMLIB(BPXPRMxx), which defines the default OMVS address space (process) region size, to 2G. This is the maximum size allowed. This

is a system-wide limit, and thus active for all z/OS UNIX address spaces. If this is not desired, then you can set the limit also just for Developer for System z in your security software.

This value can be checked and set dynamically (until the next IPL) with the following console commands, as described in *MVS System Commands* (GC28-1781):

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

Limitations stored in the security profile

Check ASSIZEMAX in the daemon'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 are 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* (GC28-1781):

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

Limitations for 64-bit addressing

Keyword MEMLIMIT in SYS1.PARMLIB(SMFPRMxx) limits how much virtual storage a 64-bit task can allocate above the 2GB bar. Unlike the REGION parameter in JCL, MEMLIMIT=0M means that the process cannot use virtual storage above the bar.

If MEMLIMIT is not specified in SMFPRMxx, the default value is 0M, so tasks are bound to the (31-bit) 2GB below the bar. The default changed in z/OS 1.10 to 2G, allowing 64-bit tasks to use up to 4GB (the 2GB below the bar and the 2GB above the bar granted by MEMLIMIT).

SYS1.PARMLIB(SMFPRMxx) values can be checked and activated with the following console commands, as described in *MVS System Commands* (GC28-1781):

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

MEMLIMIT can also be specified as parameter on an EXEC card in JCL. If no MEMLIMIT parameter is specified, the default is the value defined to SMF, except when REGION=0M is specified, in which case the default is NOLIMIT.

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 the following:
 - You do not have read authority to the FEK.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 following:
 - 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 FEK.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) APPC transaction for the TSO Commands service” on page 95 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 will not accept any connections.

The following list is a selection of Technotes currently available on the support Web site (<http://www-306.ibm.com/software/awdtools/rdz/support/>). Refer to the support Web site 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 the following 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. Check the support Web site 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.

Use the **SETOMVS LIMMSG=ALL** console command to have z/OS UNIX display console messages (BPXI040I) when any of the BPXPRMxx limits is about to be reached.

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 (such as 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 IDs (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 requisite issues

Opening MVS data sets fails

When using APPC for the TSO Commands service, reading, and writing an 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) ENTRYPOINT(BPXТУINT)
NETWORK DOMAINNAME(AF_UNIX)
        DOMAINNUMBER(1)
        MAXSOCKETS(2000)
        TYPE(UDS)
```

Another probable cause for this problem, when using APPC for the TSO Commands service, is that TCP/IP Resolver cannot resolve the host address properly due to a missing or incomplete resolver configuration file. A clear indication for this problem is the following message in lock.log:

```
clientip(0.0.0.0) <> callerip(<host IP address>)
```

Execute the fekfivpt TCP/IP IVP, as described in Chapter 7, “Installation verification,” on page 99. The resolver configuration section of the output will look like the following sample:

```
Resolver Trace Initialization Complete -> 2008/07/02 13:11:54.745964
```

```
res_init Resolver values:
Global Tcp/Ip Dataset = None
Default Tcp/Ip Dataset = None
Local Tcp/Ip Dataset = /etc/resolv.conf
Translation Table = Default
UserId/JobName = USERID
Caller API = LE C Sockets
Caller Mode = EBCDIC
```

Ensure that the definitions in the file (or data set) referenced by “Local Tcp/Ip Dataset” are correct.

This field will be blank if you do not use a default name for the IP resolver file (using the z/OS UNIX search order). If so, add the following statement to rsed.envvars, where <resolver file> or <resolver data> represents the name of your IP resolver file:

```
RESOLVER_CONFIG=<resolver file>
```

or

```
RESOLVER_CONFIG='<resolver data set>'
```

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 CAs.
- 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.

Chapter 10. Security considerations

Developer for System z provides mainframe access to users on a non-mainframe workstation. Validating connection requests, providing secure communication between the host and the workstation, and authorizing and auditing activity are therefore important aspects of the product configuration.

The security mechanisms used by Developer for System z servers and services rely on the file system it resides in being secure. This implies that only trusted system administrators should be able to update the program libraries and configuration files.

The following topics are covered in this chapter:

- “Authentication methods”
- “Connection security” on page 148
- “TCP/IP ports” on page 150
- “Using PassTickets” on page 152
- “Audit logging” on page 152
- “JES security” on page 153
- “SSL encrypted communication” on page 157
- “Client authentication using X.509 certificates” on page 158
- “Port Of Entry (POE) checking” on page 161
- “CICSTS security” on page 161
- “SCLM security” on page 162
- “Developer for System z configuration files” on page 162
- “Security definitions” on page 163

Note: Remote Systems Explorer (RSE), which provides core services such as connecting the client to the host, consists of 2 logical entities:

- RSE daemon, which manages connection setup, and is started as a started task or long running user job.
- RSE server, which handles individual client request, and is started as a thread in one or more child processes by RSE daemon.

Refer to Chapter 11, “Understanding Developer for System z,” on page 177 to learn about basic Developer for System z design concepts.

Authentication methods

Developer for System z supports multiple ways to authenticate a user ID provided by a client upon connection.

- User ID and password
- User ID and one-time password
- X.509 certificate

Note that the authentication data provided by the client is only used once, during initial connection setup. Once a user ID is authenticated, the user ID and self-generated PassTickets are used for all actions that require authentication.

User ID and password

The client provides a user ID and matching password upon connection. The user ID and password are used to authenticate the user with your security product.

User ID and one-time password

Based upon a unique token, a one-time password can be generated by a third-party product. One-time passwords improve your security setup as the unique token cannot be copied and used without the user's knowledge, and an intercepted password is useless because it is valid only once.

The client provides a user ID and the one-time password upon connection, which is used to authenticate the user ID with the security exit provided by the third party. This security exit is expected to ignore the PassTickets used to satisfy authentication requests during normal processing. The PassTickets must be processed by your security software.

X.509 certificate

A third party can provide one or more X.509 certificates that can be used for authenticating a user. When stored on secure devices, X.509 certificates combine a secure setup with ease of use for the user (no user ID or password needed).

Upon connection, the client provides a selected certificate, and optionally a selected extension, which is used to authenticate the user ID with your security product.

Note that this authentication method is only supported by the RSE daemon connection method, and that SSL must be enabled.

JES Job Monitor authentication

Client authentication is done by RSE daemon (or REXEC/SSH) as part of the client's connection request. Once the user is authenticated, self-generated PassTickets are used for all future authentication requests, including the automatic logon to JES Job Monitor.

In order for JES Job Monitor to validate the user ID and PassTicket presented by RSE, JES Job Monitor must be allowed to evaluate the PassTicket. This implies the following:

- Load module FEJMON, by default located in load library FEK.SFEKAUTH, must be APF authorized.
- Both RSE and JES Job Monitor must use the same application ID (APPLID). By default both servers use FEKAPPL as APPLID, but this can be changed by the APPLID directive in rsed.envvars for RSE and in FEJJCNFG for JES Job Monitor.

Note: Previous clients (version 7.0 and older) communicate directly with JES Job Monitor. For these connections, only the user ID and password authentication method is supported.

Connection security

Different levels of communication security are supported by RSE, which controls all communication between the client and Developer for System z services:

- External (client-host) communication can be limited to specified ports. This feature is disabled by default.

- External (client-host) communication can be encrypted using SSL. This feature is disabled by default.
- Port Of Entry (POE) checking can be used to allow host access only to trusted TCP/IP addresses. This feature is disabled by default.

Limit external communication to specified ports

The system programmer can specify the ports on which the RSE server can communicate with the client. By default, any available port is used. This range of ports has no connection with the RSE daemon port.

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

1. The client connects to host port 4035, RSE daemon.
2. The RSE daemon creates an RSE server thread.
3. The RSE server 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 RSE daemon returns the port number to the client.
5. The client connects to the host port.

Note: The process is similar for the (optional) alternative connection method using REXEC/SSH, which is described in “(Optional) Using REXEC (or SSH)” on page 93.

Communication encryption using SSL

All external Developer for System z data streams that pass through RSE can be encrypted using Secure Socket Layer (SSL). The usage of SSL is controlled by the settings in the `ssl.properties` configuration file, as described in “SSL encrypted communication” on page 157.

The Host Connect Emulator on the client connects to a TN3270 server on the host. The usage of SSL is controlled by TN3270, as documented in the *Communications Server IP Configuration Guide* (SC31-8775).

The Application Deployment Manager client uses the CICS TS Web Service or the RESTful interface to invoke the Application Deployment Manager host services. The usage of SSL is controlled by CICS TS, as documented in *RACF Security Guide for CICS TS*.

Port Of Entry checking

Developer for System z supports Port Of Entry (POE) checking, which allows host access only to trusted TCP/IP addresses. The usage of POE is controlled by the definition of specific profiles in your security software and the `enable.port.of.entry` directive in `rsed.envvars`, as described in “Port Of Entry (POE) checking” on page 161.

Note that activating POE will impact other TCPIP applications that support POE checking, such as INETD.

TCP/IP ports

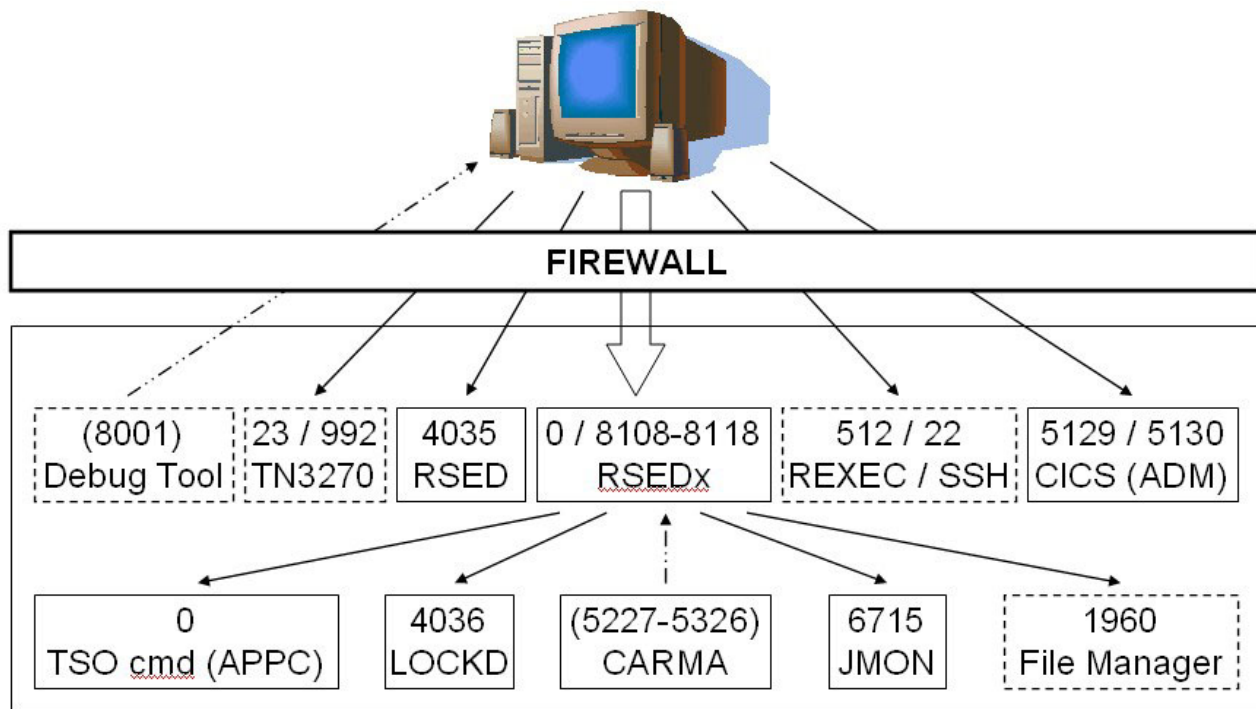


Figure 40. TCP/IP ports

Figure 40 shows the TCP/IP ports that can be used by Developer for System z. The arrows show which party does the bind (arrowhead side) and which one connects.

External communication

Define the following ports to your firewall protecting the z/OS host, as they are used for client-host communication (using the tcp protocol):

- RSE daemon for client-host communication setup, default port 4035. Communication on this port can be encrypted using SSL.
- RSE server for client-host communication. By default, any available port is used, but this can be limited to a specified range with the `_RSE_PORTRANGE` definition in `rsed.envvars`. Communication on this port can be encrypted using SSL.
- (optional) Either INETD service for remote (host-based) actions in z/OS UNIX subprojects:
 - REXEC (z/OS UNIX version), default port 512.
 - SSH (z/OS UNIX version), default port 22. Communication on this port is encrypted using SSL.
- (optional) TN3270 Telnet service for the Host Connect Emulator, default port 23. Communication 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.
- (optional) Either or both CICSTS application interfaces for Application Deployment Manager:
 - RESTful interface, default port 5130.

- Web Services interface, default port 5129. Communication on this port can be encrypted using SSL.

Note:

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

Internal communication

Several Developer for System z host services run in separate threads or address spaces and are using TCP/IP sockets as communication mechanism. All these services use RSE for communicating with the client, making their data stream confined to the host only. For some services any available port will be used, for others the system programmer can choose the port or port range that will be used:

- JES Job Monitor for JES-related services, default port 6715. The port can be set in the FEJJCNFG configuration member.
- Lock daemon for data set lock-related services, default port 4036. The port can be set in the rsed.envvars configuration member.
- (optional) File Manager Integration for interacting with IBM File Manager, default port 1960.
- (optional) CARMA communication, default port range 5227-5326 (100 ports). The port range can be set in the CRASRV.properties configuration file.
- (optional) The APPC version of the TSO Commands service uses any socket available to communicate with the lock manager (which enqueues MVS data sets for clients). You cannot set a specific port range to be used.

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

CARMA and TCP/IP ports

In most cases, like for RSE daemon, a server binds to a port and listens for connection requests. CARMA however uses a different approach, as the CARMA server is not active yet when the client initiates the connection request.

When the client sends a connection request, the CARMA miner, which is active as a user thread in a RSE thread pool, will find a free port in the range specified in the CRASRV.properties configuration file and binds to it. The miner then starts the CARMA server and passes the port number, so that the server knows to which port to connect. Once the server is connected, the client can send requests to the server and receive the results.

So from a TCP/IP perspective, RSE (via the CARMA miner) is the server that binds to the port, and the CARMA server is the client connecting to it.

Using PassTickets

After logon, PassTickets are used to establish thread security within the RSE server. This feature cannot be disabled. PassTickets are system generated passwords with a lifespan of about 10 minutes. The generated PassTickets are based upon the DES encryption algorithm, the user ID, the application ID, a time and date stamp, and a secret key. This secret key is a 64 bit number (16 hex characters) that must be defined to your security software.

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

1. The client connects to host port 4035, RSE daemon.
2. The RSE daemon authenticates the client, using the credentials presented by the client.
3. The RSE daemon creates a unique client ID and an RSE server thread.
4. The RSE server generates a PassTicket and creates a security environment for the client, using the PassTicket as password.
5. The client connects to the host port returned by the RSE daemon.
6. The RSE server validates the client using the client ID.
7. The RSE server uses a newly generated PassTicket as password for all future actions requiring a password.

The actual password of the client is no longer needed after initial authentication because SAF-compliant security products can evaluate both PassTickets and regular passwords. RSE server generates and uses a PassTicket each time a password is required, resulting in a (temporary) valid password for the client.

Using PassTickets allows RSE to set up a user-specific security environment at will, without the need of storing all user IDs and passwords in a table, which could be compromised. It also allows for client authentication methods that do not use reusable passwords, such as X.509 certificates.

Security profiles in the APPL and PTKTDATA classes are required to be able to use PassTickets. These profiles are application specific and thus do not impact your current system setup.

PassTickets being application specific implies that both RSE and JES Job Monitor must use the same application ID (APPLID). By default both servers use FEKAPPL as APPLID, but this can be changed by the APPLID directive in rsed.envvars for RSE and in FEJJCNFG for JES Job Monitor.

You should not use OMVSAPPL as application ID, because it will open the secret key to most z/OS UNIX applications. You should also not use the default MVS application ID, which is MVS followed by the system's SMF ID, because this will open the secret key to most MVS applications (including user batch jobs).

Attention: The client connection request will fail if PassTickets are not set up correctly.
--

Audit logging

Developer for System z supports audit logging of actions that are managed by the RSE daemon. The audit logs are stored as text files in the daemon log directory, using the CSV (Comma Separated Value) format.

Audit control

Multiple options in `rsed.envvars` influence the audit function, as documented in “Defining extra Java startup parameters with `_RSE_JAVAOPTS`” on page 37.

- The audit function is enabled/disabled by the `enable.audit.log` option.
- The audit defaults are controlled by the `audit.*` options.
- The location of the audit log files is controlled by the `daemon.log` option.
- The code page used for writing the audit log is controlled by the `_RSE_HOST_CODEPAGE` directive, as documented in “`rsed.envvars`, RSE configuration file” on page 28.

The **modify switch** operator command can be used to manually switch to a new audit log file, as documented in Chapter 8, “Operator commands,” on page 115.

A warning message is sent to the console when the file system holding the audit log files is running low on free space. This console message (FEK103E) is repeated regularly until the low space issue is resolved. Refer to “Console messages” on page 121 for a list of console messages generated by RSE.

Audit data

A new audit log file is started after a predetermined time or when the **modify switch** operator command is issued. The old log file is saved as `audit.log.yyyymmdd.hhmmss`, where `yyyymmdd.hhmmss` is the date/timestamp when this log was closed. The system date/timestamp assigned to the file indicates the creation of the log file. The combination of the two dates shows the time period covered by this audit log file.

The following actions are logged:

- System access (connect, disconnect)
- JES spool access (submit, display, hold, release, cancel, purge)
- Data set access (read, write, create, delete, rename, compress, migration, recall)
- Execution of TSO commands

Each logged action is stored (with a date/timestamp) using the CSV (Comma Separated Value) format, which can be read by an automation or data analysis tool.

Audit log files have permission bit mask 640 (-rw-r-----), which means that the owner (RSE daemon z/OS UNIX uid) has read and write access, and the owner's (default) group has read access. All other access attempts are denied, unless it is done by a super user (UID 0) or somebody with sufficient permission to the `SUPERUSER.FILESYS` profile in the `UNIXPRIV` class.

JES security

Developer for System z allows clients access to the JES spool through the JES Job Monitor. The server provides basic access limitations, which can be extended with the standard spool file protection features of your security product. Operator actions (Hold, Release, Cancel, and Purge) against spool files are done through an EMCS console, for which conditional permits must be set up.

Actions against jobs - target limitations

JES Job Monitor does not provide Developer for System z users full operator access to the JES spool. Only the Hold, Release, Cancel, and Purge commands are

available, and by default, only for spool files owned by the user. The commands are issued by selecting the appropriate option in the client menu structure (there is no command prompt). The scope of the commands can be widened, using security profiles to define for which jobs the commands are available.

Similar to the SDSF **SJ** action character, JES Job Monitor also supports the Show JCL command to retrieve the JCL that created the selected job output, and show it in an editor window. JES Job Monitor retrieves the JCL from JES, making it a useful function for situations in which the original JCL member is not easily located.

Table 21. JES Job Monitor console commands

Action	JES2	JES3
Hold	\$Hx(jobid) with x = {J, S or T}	*F,J=jobid,H
Release	\$Ax(jobid) with x = {J, S or T}	*F,J=jobid,R
Cancel	\$Cx(jobid) with x = {J, S or T}	*F,J=jobid,C
Purge	\$Cx(jobid),P with x = {J, S or T}	*F,J=jobid,C
Show JCL	not applicable	not applicable

The available JES commands listed in Table 21 are by default limited to jobs owned by the user. This can be changed with the LIMIT_COMMANDS directive, as documented in “FEJJCNFG, JES Job Monitor configuration file” on page 24.

Table 22. LIMIT_COMMANDS command permission matrix

LIMIT_COMMANDS	Job owner	
	User	Other
USERID (default)	Allowed	Not allowed
LIMITED	Allowed	Allowed only if explicitly permitted by security profiles
NOLIMIT	Allowed	Allowed if permitted by security profiles or when the JESSPOOL class is not active

JES uses the JESSPOOL class to protect SYSIN/SYSOUT data sets. Similar to SDSF, JES Job Monitor extends the use of the JESSPOOL class to protect job resources as well.

If LIMIT_COMMANDS is not USERID, then JES Job Monitor will query for permission to the related profile in the JESSPOOL class, as shown in the following table.

Table 23. Extended JESSPOOL profiles

Command	JESSPOOL profile	Required access
Hold	nodeid.userid.jobname.jobid	ALTER

Table 23. Extended JESSPOOL profiles (continued)

Command	JESSPOOL profile	Required access
Release	nodeid.userid.jobname.jobid	ALTER
Cancel	nodeid.userid.jobname.jobid	ALTER
Purge	nodeid.userid.jobname.jobid	ALTER
Show JCL	nodeid.userid.jobname.jobid.JCL	READ

Use the following substitutions in the preceding table:

nodeid	NJE node ID of the target JES subsystem
userid	Local user ID of the job owner
jobname	Name of the job
jobid	JES job ID

If the JESSPOOL class is not active, then there is different behavior for the LIMITED and NOLIMIT value of LIMIT_COMMANDS, as described in Table 9 on page 27. The behavior is identical when JESSPOOL is active, since the class, by default, denies permission if a profile is not defined.

Actions against jobs - execution limitations

The second phase of JES spool command security, after specifying the permitted targets, includes the permits needed to actually execute the operator command. This execution authorization is enforced by the z/OS and JES security checks.

Note that Show JCL is not an operator command such as the other JES Job Monitor commands (Hold, Release, Cancel, and Purge), so the limitations below do not apply because there is no further security check.

JES Job Monitor issues all JES operator commands requested by a user through an extended MCS (EMCS) console, whose name is controlled with the CONSOLE_NAME directive, as documented in “FEJJCENFG, JES Job Monitor configuration file” on page 24.

This setup allows the security administrator to define granular command execution permits using the OPERCMDS and CONSOLE classes.

- In order to use an EMCS console, a user must have (at least) READ authority to the MVS.MCSOPER.console-name profile in the OPERCMDS class. Note that if no profile is defined, the system will grant the authority request.
- In order to execute a JES operator command, a user must have sufficient authority to the JES%.** (or more specific) profile in the OPERCMDS class. Note that if no profile is defined, or the OPERCMDS class is not active, JES will fail the command.
- The security administrator can also require that a user must use JES Job Monitor when executing the operator command by specifying WHEN(CONSOLE(JMON)) on the PERMIT definition. The CONSOLE class must be active for this setup to work. Note that the CONSOLE class being active is sufficient; no profiles are checked for EMCS consoles.

Assuming the identity of the JES Job Monitor server by creating a JMON console from a TSO session is prevented by your security software. Even though the

console can be created, the point of entry is different (JES Job Monitor versus TSO). JES commands issued from this console will fail the security check, if your security is set up as documented in this publication and the user does not have authority to JES commands through other means.

Note that JES Job Monitor cannot create the console when a command must be executed if the console name is already in use. To prevent this, the system programmer can set the GEN_CONSOLE_NAME=ON directive in the JES Job Monitor configuration file or the security administrator can define security profiles to stop TSO users from creating a console. The following sample RACF commands prevent everyone (except those permitted) from creating a TSO or SDSF console:

- RDEFINE TSOAUTH CONSOLE UACC(NONE)
- PERMIT CONSOLE CLASS(TSOAUTH) ACCESS(READ) ID(#userid)
- RDEFINE SDSF ISFCMD.ODSP.ULOG.* UACC(NONE)
- PERMIT ISFCMD.ODSP.ULOG.* CLASS(SDSF) ACCESS(READ) ID(#userid)

Note: Without being authorized for these operator commands, users will still be able to submit jobs and read job output through the JES Job Monitor, if they have sufficient authority to possible profiles that protect these resources (such as those in the JESINPUT, JESJOBS and JESSPOOL classes).

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

Access to spool files

JES Job Monitor allows browse access to all spool files by default. This can be changed with the LIMIT_VIEW directive, as documented in "FEJJCNFG, JES Job Monitor configuration file" on page 24.

Table 24. LIMIT_VIEW browse permission matrix

LIMIT_VIEW	Job owner	
	User	Other
USERID	Allowed	Not allowed
NOLIMIT (default)	Allowed	Allowed if permitted by security profiles or when the JESSPOOL class is not active

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 the users need access to a wider range of jobs, but not all, use the standard spool file protection features of your security product, such as the JESSPOOL class.

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 1.7 (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.

SSL encrypted communication

External (client-host) communication can be encrypted using SSL (Secure Socket Layer). This feature is disabled by default and is controlled by the settings in `ssl.properties`, as documented in “(Optional) RSE SSL encryption” on page 85.

RSE daemon and RSE server support different mechanisms to store certificates due to architectural differences between the two. This implies that SSL definitions and certificates are required for both RSE daemon and RSE server. A shared certificate can be used if RSE daemon and RSE server use the same certificate management method.

Table 25. SSL certificate storage mechanisms

Certificate storage	Created and managed by	RSE daemon	RSE server
key ring	SAF compliant security product	supported	supported
key database	z/OS UNIX's gskkyman	supported	/
key store	Java's keytool	/	supported

Note: SAF-compliant key rings are the preferred method for managing certificates.

SAF-compliant key rings can store the certificate's private key either in the security database or by using ICSF (Integrated Cryptographic Service Facility), the interface to System z cryptographic hardware.

ICSF is recommended for the storage of the private keys associated with digital certificates, because it is a more secure solution than non-ICSF private key management. ICSF ensures that private keys are encrypted under the ICSF master key and that access to them is controlled by general resources in the CSFKEYS and CSFSERV security classes. In addition, operational performance is improved because ICSF utilizes the hardware Cryptographic Coprocessor.

RSE daemon uses System SSL functions to manage SSL encrypted communications. This implies that `SYS1.SIEALNKE` must be program controlled by your security software and available to RSE via `LINKLIST` or the `STEPLIB` directive in `rsd.envvars`.

The RSE user ID (`STCRSE` in the sample commands below) needs authorization to access his key ring and the related certificates when SAF-compliant key rings are used for either RSE daemon or RSE server.

- `RDEFINE FACILITY IRR.DIGTCERT.LIST UACC(NONE)`
- `RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)`
- `PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ACCESS(READ) ID(stcrse)`
- `PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ACCESS(READ) ID(stcrse)`
- `SETROPTS RACLIST(FACILITY) REFRESH`

Refer to Appendix A, “Setting up SSL and X.509 authentication,” on page 269 for more details on activating SSL for Developer for System z.

Client authentication using X.509 certificates

RSE daemon supports users authenticating themselves with an X.509 certificate. Using SSL encrypted communication is a prerequisite for this function, as it is an extension to the host authentication with a certificate used in SSL.

RSE daemon starts the client authentication process by validating the client certificate. Some key aspects that are checked are the dates the certificate is valid and the trust-worthiness of the Certificate Authority (CA) used to sign the certificate. Optionally, a (third party) Certificate Revocation List (CRL) can also be consulted.

After RSE daemon validates the certificate, it is processed for authentication. The certificate is passed on to your security product for authentication, unless `rsed.envvars.directive.enable.certificate.mapping` is set to `false`, at which point RSE daemon will do the authentication.

If successful, the authentication process will determine the user ID to be used for this session, which is then tested by RSE daemon to ensure it is usable on the host system where RSE daemon is running

The last check (which is done for every authentication mechanism, not just X.509 certificates) verifies that the user ID is allowed to use Developer for System z.

If you are familiar with the SSL security classifications used by TCP/IP, the combination of these validation steps match the “Level 3 Client authentication” specifications (the highest available).

Certificate Authority (CA) validation

Part of the certificate validation process includes checking that the certificate was signed by a Certificate Authority (CA) you trust. In order to do so, RSE daemon must have access to a certificate that identifies the CA.

When using the **gskkyman** key database for your SSL connection, the CA certificate must be added to the key database.

When using an SAF key ring (which is the advised method), you must add the CA certificate to your security database as a CERTAUTH certificate with the TRUST or HIGHTRUST attribute, as shown in this sample RACF command:

- `RACDCERT CERTAUTH ADD(dsn) HIGHTRUST WITHLABEL('label')`

Note that most security products already have the certificates for well known CA's available in their database with a NOTRUST status. Use the following sample RACF commands to list the existing CA certificates and mark one as trusted based on the label assigned to it.

- `RACDCERT CERTAUTH LIST`
- `RACDCERT CERTAUTH ALTER(LABEL('HighTrust CA')) HIGHTRUST`

Note: The HIGHTRUST status is required if you rely on RACF authenticating the user based upon the HostIdMappings extension in the certificate. Refer to “Authentication by your security software” on page 159 for more information.

Once the CA certificate is added to your security database, it must be connected to the RSE key ring, as shown in this sample RACF command:

- `RACDCERT ID(stcrse) CONNECT(CERTAUTH LABEL('HighTrust CA'))
RING(rdzssl.racf))`

Refer to *Security Server RACF Command Language Reference* (SA22-7687) for more information on the **RACDCERT** command.

Attention: If you rely on RSE daemon instead of your security software to authenticate a user you must be cautious not to mix CAs with a TRUST and HIGHTRUST status in your SAF key ring or **gskkyman** key database. RSE daemon is not able to differentiate between the two, so certificates signed by a CA with TRUST status will be valid for user ID authentication purposes.

(Optional) Query a Certificate Revocation List (CRL)

If desired, you can instruct RSE daemon to check one or more Certificate Revocation List(s) (CRL) to add extra security to the validation process. This is done by adding CRL-related environment variables to `rsed.envvars`. Refer to “`rsed.envvars`, RSE configuration file” on page 28 for information on these sample variables:

- `GSK_CRL_SECURITY_LEVEL`
- `GSK_LDAP_SERVER`
- `GSK_LDAP_PORT`
- `GSK_LDAP_USER`
- `GSK_LDAP_PASSWORD`

Refer to the *Cryptographic Services System Secure Sockets Layer Programming* (SC24-5901) for more information on these and other environment variables used by z/OS System SSL.

Note: Be careful when specifying other z/OS System SSL environment variables (`GSK_*`) in `rsed.envvars`, as they might change the way RSE daemon handles SSL connections and certificate authentication.

Authentication by your security software

RACF performs several checks to authenticate a certificate and return the associated user ID. Note that other security products might do this differently. Refer to your security product documentation for more information on the `initACEE` function used to do the authentication (query mode).

1. RACF checks if the certificate is defined in the DIGTCERT class. If so, RACF returns the user ID that was associated with this certificate when it was added to the RACF database.

Certificates are defined to RACF using the **RACDCERT** command, as in the following example:

```
RACDCERT ID(userid) ADD(dsn) TRUST WITHLABEL('label')
```

2. If the certificate is not defined, RACF checks to see if there is a matching certificate name filter defined in the DIGTNMAP or DIGTCRIT classes. If so, it returns the user ID associated with the most specific matching filter.

Note: It is advised not to use name filters for certificates used by Developer for System z, as these filters map all certificates to a single user ID. The result is that all your Developer for System z users will log on with the same user ID.

3. If there is no matching name filter, RACF locates the HostIdMappings certificate extension and extracts the embedded user ID and host name pair. If found and validated, RACF returns the user ID defined within the HostIdMappings extension.

The user ID and host name pair is valid if all these conditions are true:

- The CA certificate used to sign this certificate is marked as HIGHTRUST in the DIGTCERT class.
- The user ID stored in the extension has a valid length (1 to 8 characters).
- The user ID assigned to RSE daemon has (at least) READ authority to the IRR.HOST.hostname profile in the SERVAUTH class, where hostname is the host name stored in the extension. This is usually a domain name, such as CDFMVS08.RALEIGH.IBM.COM.

The definition of the HostIdMappings extension in ASN.1 syntax is:

```
id-ce-hostIdMappings OBJECT IDENTIFIER ::= { 1 3 18 0 2 18 1 }
HostIdMappings ::= SET OF HostIdMapping
HostIdMapping ::= SEQUENCE {
    hostName          IMPLICIT[1] IA5String,
    subjectId         IMPLICIT[2] IA5String,
    proofOfIdPossession IdProof OPTIONAL
}
IdProof ::= SEQUENCE {
    secret            OCTET STRING,
    encryptionAlgorithm OBJECT IDENTIFIER
}
```

Note: A HostIdMappings extension is not honored if the target user ID was created after the start of the validity period for the certificate containing the HostIdMappings extension. Therefore, if you are creating user IDs specifically for certificates with HostIdMappings extensions, make sure that you create the user IDs before the certificate requests are submitted.

Refer to *Security Server RACF Security Administrator's Guide* (SA22-7683) for more information on X.509 certificates, how they are managed by RACF, and how to define certificate name filters. Refer to *Security Server RACF Command Language Reference* (SA22-7687) for more information on the **RACDCERT** command.

Authentication by RSE daemon

Developer for System z can do basic X.509 certificate authentication without relying on your security product. Authentication done by RSE daemon requires a user ID and host name to be defined in a certificate extension, and is only activated if the enable.certificate.mapping directive in rsed.envvars is set to FALSE.

This function is intended to be used if your security product does not support authenticating a user based upon an X.509 certificate, or if your certificate would fail the test(s) done by your security product (for example, the certificate has a faulty identifier for the HostIdMappings extension and there is no name filter or definition in DIGTCERT).

The client will query the user for the extension identifier (OID) to use, which is by default the HostIdMappings OID, {1 3 18 0 2 18 1}.

RSE daemon will extract the user ID and host name from it using the format of the HostIdMappings extension. This format is described in "Authentication by your security software" on page 159.

The user ID and host name pair is valid if all these conditions are true:

- The user ID stored in the extension has a valid length (1 to 8 characters).
- The user ID assigned to RSE daemon has (at least) READ authority to the IRR.HOST.hostname profile in the SERVAUTH class, where hostname is the host name stored in the extension. This is usually a domain name, such as CDFMVS08.RALEIGH.IBM.COM.

Attention: It is up to the security administrator to ensure that all CAs known to RSE daemon are highly trusted, because RSE daemon cannot check if the one who signed the client certificate is highly trusted or just trusted. See “Certificate Authority (CA) validation” on page 158 for more information on accessible CA certificates.

Port Of Entry (POE) checking

Developer for System z supports Port Of Entry (POE) checking, which allows host access only to trusted TCP/IP addresses. This feature is disabled by default and requires the definition of the BPX.POE security profile, as shown in the following sample RACF commands:

- RDEFINE FACILITY BPX.POE UACC(NONE)
- PERMIT BPX.POE CLASS(FACILITY) ACCESS(READ) ID(STCRSE)
- SETROPTS RACLIST(FACILITY) REFRESH

Note:

- RSE must be configured to use POE by uncommenting the “enable.port.of.entry=true” option in rsed.envvars, as documented in “Defining extra Java startup parameters with _RSE_JAVAOPTS” on page 37.
- The RSE user ID STCRSE requires UID(0) when this profile is not defined and POE checking is enabled in rsed.envvars.
- Defining BPX.POE will impact other TC/PIP applications that support POE checking, such as INETD.
- Security zones (EZB.NETACCESS.** profiles, which are IP address ranges) should be set up in the SERVAUTH class to use the full strength of POE checking.

Refer to *Communications Server IP Configuration Guide* (SC31-8775) for more information on network access control using POE checking.

CICSTS security

Developer for System z allows, through Application Deployment Manager, CICS administrators to control which CICS resource definitions are editable by the developer, their default values, and the display of a CICS resource definition by means of the CICS Resource Definition (CRD) server. Refer to Chapter 15, “CICSTS considerations,” on page 231 for more information on the required CICS TS security definitions.

CRD repository

The CRD server repository VSAM data set holds all the default resource definitions and must therefore be protected against updates, but developers must be allowed to read the values stored here.

CICS transactions

Developer for System z supplies multiple transactions that are used by the CRD server when defining and inquiring CICS resources. When the transaction is attached, CICS resource security checking, if enabled, insures that the user ID is authorized to run the transaction ID.

SSL encrypted communication

The Application Deployment Manager client uses CICS TS Web Services or the RESTful interface to invoke the CRD server. The usage of SSL for this communication is controlled by the CICS TS TCPIP SERVICE definition, as documented in and *RACF Security Guide for CICS TS*.

SCLM security

The SCLM Developer Toolkit service offers optional security functionality for the Build, Promote, and Deploy functions.

If security is enabled for a function by the SCLM administrator, SAF calls are made to verify authority to execute the protected function with the caller's or a surrogate user ID.

Refer to *SCLM Developer Toolkit Administrator's Guide* (SC23-9801), for more information on the required SCLM security definitions.

Developer for System z configuration files

There are several Developer for System z configuration files whose directives impact the security setup. Based upon the information in this chapter, the security administrator and systems programmer can decide what the settings should be for the following directives.

JES Job Monitor - FEJJC�FG

- `LIMIT_COMMANDS={USERID | LIMITED | NOLIMIT}`
Define against which jobs actions can be done (excluding browse and submit). For more information, see "Actions against jobs - target limitations" on page 153.
- `LIMIT_VIEW={USERID | NOLIMIT}`
Define which spool files can be browsed. For more information, see "Access to spool files" on page 156.
- `APPLID={FEKAPPL | *}`
Application ID used for PassTicket creation/validation. For more information, see "Using PassTickets" on page 152.

Note: Details on these and other FEJJC�FG directives are available in "FEJJC�FG, JES Job Monitor configuration file" on page 24.

RSE - rsed.envvars

- `(_RSE_JAVAOPTS) -DDENY_PASSWORD_SAVE={true | false}`
Deny users to save their host password on the client. For more information, see "Defining extra Java startup parameters with _RSE_JAVAOPTS" on page 37.
- `(_RSE_JAVAOPTS) -DDSTORE_IDLE_SHUTDOWN_TIMEOUT=value`
Timer to disconnect idle clients. For more information, see "Defining extra Java startup parameters with _RSE_JAVAOPTS" on page 37.

- (`_RSE_JVAOPTS`) `-DAPPLID={FEKAPPL | *}`
Application ID used for PassTicket creation/validation. For more information, see “Using PassTickets” on page 152.
- (`_RSE_JVAOPTS`) `-Denable.port.of.entry={true | false}`
Enable Port Of Entry checking. For more information, see “Port Of Entry (POE) checking” on page 161.
- (`_RSE_JVAOPTS`) `-Denable.certificate.mapping={true | false}`
Use your security product to authenticate users with an X.509 certificate. For more information, see “Client authentication using X.509 certificates” on page 158.
- (`_RSE_JVAOPTS`) `-Ddaemon.log={/var/rdz/logs | *}`
Location of the audit log files. For more information, see “Audit logging” on page 152.

Note: Details on these and other `rsed.envvars` directives are available in “`rsed.envvars`, RSE configuration file” on page 28.

RSE - `ssl.properties`

- `daemon_keydb_file={SAF key ring name | gskkyman key database name}`
Location of the RSE daemon certificate. For more information, see “SSL encrypted communication” on page 157.
- `daemon_key_label=certificate label`
Name of the RSE daemon certificate. For more information, see “SSL encrypted communication” on page 157.
- `server_keystore_file={SAF key ring name | Java key store name}`
Location of the RSE server certificate. For more information, see “SSL encrypted communication” on page 157.
- `server_keystore_label=certificate label`
Name of the RSE server certificate. For more information, see “SSL encrypted communication” on page 157.
- `server_keystore_type={JKS | JCERACFKS | JCECCARACFKS}`
Type of key store used (Java key store or SAF key ring). For more information, see “SSL encrypted communication” on page 157.

Note: Details on these and other `ssl.properties` directives are available in “(Optional) RSE SSL encryption” on page 85.

Security definitions

Customize and submit sample member FEKRACF, which has sample RACF and z/OS UNIX commands to create the basic security definitions for Developer for System z.

FEKRACF is located in `FEK.#CUST.JCL`, unless you specified a different location when you customized and submitted job `FEK.SFEKSAMP(FEKSETUP)`. See “Customization setup” on page 13 for more details.

Refer to the *RACF Command Language Reference* (SA22-7687), for more information on RACF commands.

Note:

- For those sites that use CA ACF2™ for z/OS, please refer to the following link, <https://support.ca.com/irj/portal/kbtech?ipLogNrow=0&docid=492389&searchID=TEC492389>, for details on the security commands necessary to properly configure Developer for System z.
- For those sites that use CA Top Secret® for z/OS, please refer to your product page on the CA support site (<https://support.ca.com>) and check for the related Developer for System z Knowledge Document. This Knowledge Document has details on the security commands necessary to properly configure Developer for System z.

The following sections describe the required steps, optional configuration and possible alternatives.

Requirements and checklist

To complete the security setup, the security administrator needs to know the values listed in Table 26. These values were defined during previous steps of the installation and customization of Developer for System z.

Table 26. Security setup variables

Description	<ul style="list-style-type: none"> • Default value • Where to find the answer 	Value
Developer for System z product high level qualifier	<ul style="list-style-type: none"> • FEK • SMP/E installation 	
Developer for System z customization high level qualifier	<ul style="list-style-type: none"> • FEK.#CUST • FEK.SFEKSAMP(FEKSETUP), as described in “Customization setup” on page 13. 	
JES Job Monitor started task name	<ul style="list-style-type: none"> • JMON • FEK.#CUST.PROCLIB(JMON), as described in “PROCLIB changes” on page 19. 	
RSE daemon started task name	<ul style="list-style-type: none"> • RSED • FEK.#CUST.PROCLIB(RSED), as described in “PROCLIB changes” on page 19. 	
Lock daemon started task name	<ul style="list-style-type: none"> • LOCKD • FEK.#CUST.PROCLIB(LOCKD), as described in “PROCLIB changes” on page 19. 	
Application ID	<ul style="list-style-type: none"> • FEKAPPL • /etc/rdz/rsed.envvars, as described in “Defining extra Java startup parameters with _RSE_JAVAOPTS” on page 37 	

The following list is an overview of the required actions to complete the basic security setup of Developer for System z. As documented in the sections below, different methods can be used to fulfill these requirements, depending on the

desired security level. Refer to the previous sections for information on the security setup of optional Developer for System z services.

- “Activate security settings and classes”
- “Define an OMVS segment for Developer for System z users” on page 166
- “Define data set profiles” on page 166
- “Define the Developer for System z started tasks” on page 169
- “Define JES command security” on page 170
- “Define RSE as a secure z/OS UNIX server” on page 171
- “Define MVS program controlled libraries for RSE” on page 172
- “Define application protection for RSE” on page 172
- “Define PassTicket support for RSE” on page 173
- “Define z/OS UNIX program controlled files for RSE” on page 174
- “Verify security settings” on page 174

Activate security settings and classes

Developer for System z utilizes a variety of security mechanisms to ensure a secure and controlled host environment for the client. In order to do so, several classes and security settings must be active, as shown with the following sample RACF commands:

- Display current settings
 - SETROPTS LIST
- Activate facility class for z/OS UNIX and digital certificate profiles
 - SETROPTS GENERIC(FACILITY)
 - SETROPTS CLASSACT(FACILITY) RACLIST(FACILITY)
- Activate started task definitions
 - SETROPTS GENERIC(STARTED)
 - RDEFINE STARTED ** STDATA(USER(=MEMBER) GROUP(STCGROUP) TRACE(YES))
 - SETROPTS CLASSACT(STARTED) RACLIST(STARTED)
- Activate console security for JES Job Monitor
 - SETROPTS GENERIC(CONSOLE)
 - SETROPTS CLASSACT(CONSOLE) RACLIST(CONSOLE)
- Activate operator command protection for JES Job Monitor
 - SETROPTS GENERIC(OPERCMDS)
 - SETROPTS CLASSACT(OPERCMDS) RACLIST(OPERCMDS)
- Activate application protection for RSE
 - SETROPTS GENERIC(APPL)
 - SETROPTS CLASSACT(APPL) RACLIST(APPL)
- Activate secured signon using PassTickets for RSE
 - SETROPTS GENERIC(PKTCDATA)
 - SETROPTS CLASSACT(PKTCDATA) RACLIST(PKTCDATA)
- Activate program control to ensure that only trusted code can be loaded by RSE
 - RDEFINE PROGRAM ** ADDMEM('SYS1.CMDLIB'//NOPADCHK) UACC(READ)
 - SETROPTS WHEN(PROGRAM)

Note: Do not create the ** profile if you already have a * profile in the PROGRAM class. It obscures and complicates the search path used by your security software. In this case, you must merge the existing * and the new ** definitions. IBM recommends to use the ** profile, as documented in *Security Server RACF Security Administrator's Guide* (SA22-7683).

Attention: Some products, such as FTP, require being program controlled if "WHEN PROGRAM" is active. Test this before activating it on a production system.

- (Optional) Activate X.509 HostIdMappings and extended Port Of Entry (POE) support
 - SETROPTS GENERIC(SERVAUTH)
 - SETROPTS CLASSACT(SERVAUTH) RACLIST(SERVAUTH)

Define an OMVS segment for Developer for System z users

A RACF OMVS segment (or equivalent) that specifies a valid non-zero z/OS UNIX user ID (UID), home directory, and shell command must be defined for each user of Developer for System z. Their default group also requires an OMVS segment with a group id.

Replace in the following sample RACF commands the #userid, #user-identifier, #group-name and #group-identifier placeholders with actual values:

- ALTUSER #userid
OMVS(UID(#user-identifier) HOME(/u/#userid) PROGRAM(/bin/sh) NOASSIZEMAX)
- ALTGROUP #group-name OMVS(GID(#group-identifier))

Although it is advised not to do so, you can use the shared OMVS segment defined in the BPX.DEFAULT.USER profile of the FACILITY class to fulfill the OMVS segment requirement for Developer for System z.

Define data set profiles

READ access for users and ALTER for system programmers suffices for most Developer for System z data sets. Replace the #sysprog placeholder with valid user IDs or RACF group names. Also ask the system programmer who installed and configured the product for the correct data set names. FEK is the default high-level qualifier used during installation and FEK.#CUST is the default high-level qualifier for data sets created during the customization process.

- ADDGROUP (FEK) OWNER(IBMUSER) SUPGROUP(SYS1)
DATA('RATIONAL DEVELOPER FOR SYSTEM Z - HLQ STUB')
- ADDSD 'FEK.*.*' UACC(READ)
DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
- PERMIT 'FEK.*.*' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
- SETROPTS GENERIC(DATASET) REFRESH

Note:

- You are strongly advised to protect FEK.SFEKAUTH against updates since this data set is APF authorized. The same is true for FEK.SFEKLOAD and FEK.SFEKLPA, but here because these data sets are program controlled.
- The sample commands in this publication and in the FEKRACF job assume that EGN (Enhanced Generic Naming) is active. This allows the usage of the ** qualifier to represent any number of qualifiers in the DATASET class. Substitute ** with * if EGN is not active on your system. Refer to *Security Server RACF Security Administrator's Guide* (SA22-7683) for more information on EGN.

Some of the optional Developer for System z components require additional security data set profiles. Replace the #sysprog, #ram-developer and #cicsadmin placeholders with valid user ID's or RACF group names:

- If SCLM Developer Toolkit's long/short name translation is used, then users require UPDATE access to the mapping VSAM, FEK.#CUST.LSTRANS.FILE.

- ADDSD 'FEK.#CUST.LSTRANS.*.*' UACC(UPDATE)
 - DATA('RATIONAL DEVELOPER FOR SYSTEM Z - SCLMDT')
 - PERMIT 'FEK.#CUST.LSTRANS.*.*' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
 - SETROPTS GENERIC(DATASET) REFRESH
- CARMA RAM (Repository Access Manager) developers require UPDATE access to the CARMA VSAMs, FEK.#CUST.CRA*.
 - ADDSD 'FEK.#CUST.CRA*.*' UACC(READ)
 - DATA('RATIONAL DEVELOPER FOR SYSTEM Z - CARMA')
 - PERMIT 'FEK.#CUST.CRA*.*' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
 - PERMIT 'FEK.#CUST.CRA*.*' CLASS(DATASET) ACCESS(UPDATE) ID(#ram-developer)
 - SETROPTS GENERIC(DATASET) REFRESH
- If Application Deployment Manager's CRD server (CICS Resource Definition) is used, then CICS administrators require UPDATE access to the CRD repository VSAM.
 - ADDSD 'FEK.#CUST.ADNREP*.*' UACC(READ)
 - DATA('RATIONAL DEVELOPER FOR SYSTEM Z - ADN')
 - PERMIT 'FEK.#CUST.ADNREP*.*' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
 - PERMIT 'FEK.#CUST.ADNREP*.*' CLASS(DATASET) ACCESS(UPDATE) ID(#cicsadmin)
 - SETROPTS GENERIC(DATASET) REFRESH
- If Application Deployment Manager's manifest repository is defined, then all CICS Transaction Server users require UPDATE access to the manifest repository VSAM.
 - ADDSD 'FEK.#CUST.ADNMAN*.*' UACC(UPDATE)
 - DATA('RATIONAL DEVELOPER FOR SYSTEM Z - ADN')
 - PERMIT 'FEK.#CUST.ADNMAN*.*' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
 - SETROPTS GENERIC(DATASET) REFRESH

Use the following sample RACF commands for a more secure setup where READ access is also controlled.

- uacc(none) data set protection
 - ADDGROUP (FEK)
 - DATA('RATIONAL DEVELOPER FOR SYSTEM Z - HLQ STUB')
 - OWNER(IBMUSER) SUPGROUP(SYS1)"
 - ADDSD 'FEK.*.*' UACC(NONE)
 - DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
 - ADDSD 'FEK.SFEKAUTH' UACC(NONE)
 - DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
 - ADDSD 'FEK.SFEKLOAD' UACC(NONE)
 - DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
 - ADDSD 'FEK.SFEKPROC' UACC(NONE)
 - DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
 - ADDSD 'FEK.#CUST.PARMLIB' UACC(NONE)
 - DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
 - ADDSD 'FEK.#CUST.CNTL' UACC(NONE)
 - DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
 - ADDSD 'FEK.#CUST.LSTRANS.*.*' UACC(NONE)
 - DATA('RATIONAL DEVELOPER FOR SYSTEM Z - SCLMDT')
 - ADDSD 'FEK.#CUST.CRA*.*' UACC(NONE)
 - DATA('RATIONAL DEVELOPER FOR SYSTEM Z - CARMA')
 - ADDSD 'FEK.#CUST.ADNREP*.*' UACC(NONE)
 - DATA('RATIONAL DEVELOPER FOR SYSTEM Z - ADN')
 - ADDSD 'FEK.#CUST.ADNMAN*.*' UACC(NONE)
 - DATA('RATIONAL DEVELOPER FOR SYSTEM Z - ADN')
- permit system programmer to manage all libraries
 - PERMIT 'FEK.*.*' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
 - PERMIT 'FEK.SFEKAUTH' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
 - PERMIT 'FEK.SFEKLOAD' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
 - PERMIT 'FEK.SFEKLOAD' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)

- PERMIT 'FEK.SFEKLOAD' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
- PERMIT 'FEK.SFEKPROC' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
- PERMIT 'FEK.#CUST.PARMLIB' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
- PERMIT 'FEK.#CUST.CNTL' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
- PERMIT 'FEK.#CUST.LSTRANS.*.*' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
- PERMIT 'FEK.#CUST.CRA*.*' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
- PERMIT 'FEK.#CUST.ADNREP*.*' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
- PERMIT 'FEK.#CUST.ADNMAN*.*' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
- permit clients to access the load and exec libraries
 - PERMIT 'FEK.SFEKAUTH' CLASS(DATASET) ACCESS(READ) ID(*)
 - PERMIT 'FEK.SFEKLOAD' CLASS(DATASET) ACCESS(READ) ID(*)
 - PERMIT 'FEK.SFEKPROC' CLASS(DATASET) ACCESS(READ) ID(*)
 - PERMIT 'FEK.#CUST.CNTL' CLASS(DATASET) ACCESS(READ) ID(*)

Note: No permits are needed for FEK.SFEKLPA, as all code that resides in LPA is accessible by everyone.

- permit JES Job Monitor to access the load & parameter library
 - PERMIT 'FEK.SFEKAUTH' CLASS(DATASET) ACCESS(READ) ID(STCJMON)
 - PERMIT 'FEK.#CUST.PARMLIB' CLASS(DATASET) ACCESS(READ) ID(STCJMON)
- (optional) permit clients to update the long/short name translation VSAM for SCLMDT
 - PERMIT 'FEK.#CUST.LSTRANS.*.*' CLASS(DATASET) ACCESS(UPDATE) ID(*)
- (optional) permit RAM developers to update the CARMA VSAMs for CARMA
 - PERMIT 'FEK.#CUST.CRA*.*' CLASS(DATASET) ACCESS(UPDATE) ID(#ram-developer)
- (optional) permit CICS users to read the CRD repository VSAM for Application Deployment Manager
 - PERMIT 'FEK.#CUST.ADNREP*.*' CLASS(DATASET) ACCESS(READ) ID(*)
- (optional) permit CICS administrators to update the CRD repository VSAM for Application Deployment Manager
 - PERMIT 'FEK.#CUST.ADNREP*.*' CLASS(DATASET) ACCESS(UPDATE) ID(#cicsadmin)
- (optional) permit CICS users to update the manifest repository VSAM for Application Deployment Manager
 - PERMIT 'FEK.#CUST.ADNMAN*.*' CLASS(DATASET) ACCESS(UPDATE) ID(*)
- (optional) permit CICS TS server to access the load library for bidi and Application Deployment Manager
 - PERMIT 'FEK.SFEKLOAD' CLASS(DATASET) ACCESS(READ) ID(#cicsts)
- (optional) permit DB2 server to access the exec library for DB2 stored procedure builder
 - PERMIT 'FEK.SFEKPROC' CLASS(DATASET) ACCESS(READ) ID(#db2)
- activate security profiles
 - SETROPTS GENERIC(DATASET) REFRESH

When controlling READ access to system data sets, you must provide Developer for System z servers and users permission to READ the following data sets:

- CEE.SCEERUN
- CEE.SCEERUN2
- CBC.SCLBDLL
- ISP.SISPLoad
- ISP.SISPLPA
- SYS1.LINKLIB
- SYS1.SIEALNKE

- REXX.V1R4M0.SEAGLPA

Note: When you use the Alternate Library for REXX product package, the default REXX runtime library name is REXX.*.SEAGALT. instead of REXX.*.SEAGLPA, as used in the sample above.

Define the Developer for System z started tasks

The following sample RACF commands create the JMON, RSED, and LOCKD started tasks, with protected user IDs (STCJMON, STCRSE, and STCLOCK respectively) and group STCGROUP assigned to them. Replace the #group-id and #user-id-* placeholders with valid OMVS IDs.

- ADDGROUP STCGROUP OMVS(GID(#group-id))
DATA('GROUP WITH OMVS SEGMENT FOR STARTED TASKS')
- ADDUSER STCJMON DFLTGROUP(STCGROUP) NOPASSWORD NAME('RDZ - JES JOBMONITOR')
OMVS(UID(#user-id-jmon) HOME(/tmp) PROGRAM(/bin/sh) NOASSIZEMAX
NOTHREADSMAX)
DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
- ADDUSER STCRSE DFLTGROUP(STCGROUP) NOPASSWORD NAME('RDZ - RSE DAEMON')
OMVS(UID(#user-id-rse) HOME(/tmp) PROGRAM(/bin/sh) ASSIZEMAX(2147483647)
NOTHREADSMAX)
DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
- ADDUSER STCLOCK DFLTGROUP(STCGROUP) NOPASSWORD NAME('RDZ - LOCK DAEMON')
OMVS(UID(#user-id-lock) HOME(/tmp) PROGRAM(/bin/sh) NOASSIZEMAX)
NOTHREADSMAX)
DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
- RDEFINE STARTED JMON.* DATA('RDZ - JES JOBMONITOR')
STDATA(USER(STCJMON) GROUP(STCGROUP) TRUSTED(NO))
- RDEFINE STARTED RSED.* DATA('RDZ - RSE DAEMON')
STDATA(USER(STCRSE) GROUP(STCGROUP) TRUSTED(NO))
- RDEFINE STARTED LOCKD.* DATA('RDZ - LOCK DAEMON')
STDATA(USER(STCLOCK) GROUP(STCGROUP) TRUSTED(NO))
- SETROPTS RACLIST(STARTED) REFRESH

Notes:

1. Ensure that the started tasks user IDs are protected by specifying the NOPASSWORD keyword.
2. Ensure that RSE server has a unique OMVS uid due to the z/OS UNIX related privileges granted to this uid.
3. RSE daemon requires a large address space size (2GB) for proper operation. You should set this value in the ASSIZEMAX variable of the OMVS segment for user ID STCRSE. This to ensure that RSE daemon will get the required region size, regardless of changes to MAXASSIZE in SYS1.PARMLIB(BPXPRMxx).
4. RSE also requires a large number of threads for proper operation. You can set the limit in the THREADSMAX variable of the OMVS segment for user ID STCRSE. This ensures that RSE will get the required thread limit, regardless of changes to MAXTHREADS or MAXTHREADTASKS in SYS1.PARMLIB(BPXPRMxx). Refer to Chapter 13, "Tuning considerations," on page 195 to determine the correct value for the thread limit.
5. User ID STCJMON is another good candidate for setting THREADSMAX in the OMVS segment, because JES Job Monitor uses a thread per client connection.

You might want to consider making the STCRSE user ID restricted. Users with the RESTRICTED attribute cannot access protected (MVS) resources they are not specifically authorized to access.

```
ALTUSER STCRSE RESTRICTED
```

To ensure that restricted users do not gain access to z/OS UNIX file system resources through the “other” permission bits, you must define the `RESTRICTED.FILESYS.ACCESS` profile in the `UNIXPRIV` class with `UACC(NONE)`. Refer to *Security Server RACF Security Administrator’s Guide* (SA22-7683) for more information on restricting user IDs.

Attention: If you use restricted user IDs, you must explicitly add the permission to access a resource with the TSO **PERMIT** or the z/OS UNIX **setfacl** commands. This includes resources where the Developer for System z documentation uses `UACC` (such as the `**` profile in the `PROGRAM` class) or where it relies on common z/OS UNIX conventions (such as everyone having read and execute permission for Java libraries). Test this before activating it on a production system.

Define JES command security

JES Job Monitor issues all JES operator commands requested by a user through an extended MCS (EMCS) console, whose name is controlled with the `CONSOLE_NAME` directive, as documented in “FEJJCNFG, JES Job Monitor configuration file” on page 24.

The following sample RACF commands give Developer for System z users conditional access to a limited set of JES commands (Hold, Release, Cancel, and Purge). Users only have execution permission if they issue the commands through JES Job monitor. Replace the `#console` placeholder with the actual console name.

- `RDEFINE OPERCMDS MVS.MCSOPER.#console UACC(READ)`
`DATA('RATIONAL DEVELOPER FOR SYSTEM Z')`
- `RDEFINE OPERCMDS JES%.** UACC(NONE)`
- `PERMIT JES%.** CLASS(OPERCMDS) ACCESS(UPDATE) WHEN(CONSOLE(JMON)) ID(*)`
- `SETOPTS RACLIST(OPERCMDS) REFRESH`

Note:

- Usage of the console is permitted if no `MVS.MCSOPER.#console` profile is defined
- The `CONSOLE` class must be active for `WHEN(CONSOLE(JMON))` to work, but there is no actual profile check in the `CONSOLE` class for EMCS consoles.
- Do not replace `JMON` with the actual console name in the `WHEN(CONSOLE(JMON))` clause. The `JMON` keyword represents the point-of-entry application, not the console name.

Attention: 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.

Table 27 and Table 28 on page 171 show the operator commands issued for JES2 and JES3, and the discrete security profiles that can be used to protect them.

Table 27. JES2 Job Monitor operator commands

Action	Command	OPERCMDS profile	Required access
Hold	<code>\$Hx(jobid)</code> with <code>x = {J, S or T}</code>	<code>jesname.MODIFYHOLD.BAT</code> <code>jesname.MODIFYHOLD.STC</code> <code>jesname.MODIFYHOLD.TSU</code>	UPDATE
Release	<code>\$Ax(jobid)</code> with <code>x = {J, S or T}</code>	<code>jesname.MODIFYRELEASE.BAT</code> <code>jesname.MODIFYRELEASE.STC</code> <code>jesname.MODIFYRELEASE.TSU</code>	UPDATE

Table 27. JES2 Job Monitor operator commands (continued)

Action	Command	OPERCMDS profile	Required access
Cancel	\$Cx(jobid) with x = {J, S or T}	jesname.CANCEL.BAT jesname.CANCEL.STC jesname.CANCEL.TSU	UPDATE
Purge	\$Cx(jobid),P with x = {J, S or T}	jesname.CANCEL.BAT jesname.CANCEL.STC jesname.CANCEL.TSU	UPDATE

Table 28. JES3 Job Monitor operator commands

Action	Command	OPERCMDS profile	Required access
Hold	*F,J=jobid,H	jesname.MODIFY.JOB	UPDATE
Release	*F,J=jobid,R	jesname.MODIFY.JOB	UPDATE
Cancel	*F,J=jobid,C	jesname.MODIFY.JOB	UPDATE
Purge	*F,J=jobid,C	jesname.MODIFY.JOB	UPDATE

Note:

- The Hold, Release, Cancel, and Purge JES operator commands, and the Show JCL command, can only be executed against spool files owned by the client user ID, unless LIMIT_COMMANDS= with value LIMITED or NOLIMIT is specified in the JES Job Monitor configuration file. Refer to “Actions against jobs - target limitations” on page 153 for more information on this.
- Users can browse any spool file, unless LIMIT_VIEW=USERID is defined in the JES Job Monitor configuration file. Refer to “Access to spool files” on page 156 for more information on this.
- Without being authorized for these operator commands, users will still be able to submit jobs and read job output through JES Job Monitor, if they have sufficient authority to possible profiles that protect these resources (such as those in the JESINPUT, JESJOBS and JESSPOOL classes).

Assuming the identity of the JES Job Monitor server by creating a JMON console from a TSO session is prevented by your security software. Even though the console can be created, the point of entry is different (JES Job Monitor versus TSO). JES commands issued from this console will fail the security check, if your security is set up as documented in this publication and the user does not have authority to the JES commands through other means.

Define RSE as a secure z/OS UNIX server

RSE requires UPDATE access to the BPX.SERVER profile to create/delete the security environment for the client’s thread. If this profile is not defined, UID(0) is required for RSE.

- RDEFINE FACILITY BPX.SERVER UACC(NONE)
- PERMIT BPX.SERVER CLASS(FACILITY) ACCESS(UPDATE) ID(STCRSE)
- SETROPTS RACLIST(FACILITY) REFRESH

Attention: Defining the BPX.SERVER profile makes z/OS UNIX as a whole switch from UNIX level security to z/OS UNIX level security, which is more secure. This might impact other z/OS UNIX applications and operations. Test this before activating it on a production system. Refer to *UNIX System Services Planning* (GA22-7800) for more information on the different security levels.

Define MVS program controlled libraries for RSE

Servers with authority to BPX.SERVER must run in a clean, program-controlled environment. This implies that all programs called by RSE must also be program controlled. For MVS load libraries, program control is managed by your security software.

RSE uses system (SYS1.LINKLIB), Language Environment's runtime (CEE.SCEERUN*) and ISPF's TSO/ISPF Client Gateway (ISP.SISPLOAD) load library.

- RALTER PROGRAM ** UACC(READ) ADDMEM('SYS1.LINKLIB'//NOPADCHK)
- RALTER PROGRAM ** UACC(READ) ADDMEM('CEE.SCEERUN'//NOPADCHK)
- RALTER PROGRAM ** UACC(READ) ADDMEM('CEE.SCEERUN2'//NOPADCHK)
- RALTER PROGRAM ** UACC(READ) ADDMEM('ISP.SISPLOAD'//NOPADCHK)
- SETROPTS WHEN(PROGRAM) REFRESH

Note: Do not use the ** profile if you already have a * profile in the PROGRAM class. It obscures and complicates the search path used by your security software. In this case, you must merge the existing * and the new ** definitions. IBM recommends using the ** profile, as documented in *Security Server RACF Security Administrator's Guide* (SA22-7683).

The following additional (prerequisite) libraries must be made program controlled to support the use of optional services. This list does not include data sets that are specific to a product that Developer for System z interacts with, such as IBM Debug Tool.

- Alternate REXX runtime library (for SCLM Developer Toolkit)
 - REXX.*.SEAGALT
- System load library (for SSL encryption)
 - SYS1.SIEALNKE
- File Manager listener load library (for File Manager integration)
 - FMN.SFMNMODA

Note: Libraries that are designed for LPA placement also require program control authorizations if they are accessed through LINKLIST or STEPLIB. This publication documents the usage of the following LPA libraries:

- ISPF (for TSO/ISPF Client Gateway)
 - ISP.SISPLPA
- REXX runtime library (for SCLM Developer Toolkit)
 - REXX.*.SEAGLPA
- Developer for System z (for CARMA)
 - FEK.SFEKLPA

Define application protection for RSE

During client logon, RSE daemon verifies that a user is allowed to use the application.

- RDEFINE APPL FEKAPPL UACC(READ) DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
- SETROPTS RACLIST(APPL) REFRESH

Note:

As described in more detail in “Define PassTicket support for RSE,” RSE supports the usage of an application ID other than FEKAPPL. The APPL class definition must match the actual application ID used by RSE.

Attention: The client connection request will fail if the application profile is not defined, or when the user lacks READ access to the profile.

Define PassTicket support for RSE

The client’s password (or other means of identification, such as an X.509 certificate) is only used to verify his identity upon connection. Afterwards, PassTickets are used to maintain thread security.

PassTickets are system-generated passwords with a lifespan of about 10 minutes. The generated PassTickets are based upon a secret key. This key is a 64 bit number (16 hex characters). Replace in the sample RACF commands below the key16 placeholder with a user-supplied 16 character hex string (characters 0-9 and A-F).

- RDEFINE PTKTDATA FEKAPPL UACC(NONE) SSIGNON(KEYMASKED(key16))
APPLDATA('NO REPLAY PROTECTION – DO NOT CHANGE')
DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
- RDEFINE PTKTDATA IRRPTAUTH.FEKAPPL.* UACC(NONE)
DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
- PERMIT IRRPTAUTH.FEKAPPL.* CLASS(PTKTDATA) ACCESS(UPDATE) ID(STCRSE)
- SETROPTS RACLIST(PTKTDATA) REFRESH

RSE supports the usage of an application ID other than FEKAPPL. Uncomment and customize the "APPLID=FEKAPPL" option in rsed.envvars to activate this, as documented in “Defining extra Java startup parameters with _RSE_JAVAOPTS” on page 37. The PTKTDATA class definitions must match the actual application ID used by RSE.

You should not use OMVSAPPL as application ID, because it will open the secret key to most z/OS UNIX applications. You should also not use the default MVS application ID, which is MVS followed by the system’s SMF ID, because this will open the secret key to most MVS applications (including user batch jobs).

Note:

- If the PTKTDATA class is already defined, verify that it is defined as a generic class before creating the profiles listed above. The support for generic characters in the PTKTDATA class is new since z/OS release 1.7, with the introduction of a Java interface to PassTickets.
- Substitute the wildcard (*) in the IRRPTAUTH.FEKAPPL.* definition with a valid user ID mask to limit the user IDs for which RSE can generate a PassTicket.
- Depending on your RACF settings, the user defining a profile may also be on the access list of the profile. It is advised that you remove this permission for the PTKTDATA profiles.
- JES Job Monitor and RSE must have the same application ID to allow JES Job Monitor to evaluate the PassTickets presented by RSE.
- If the system has a cryptographic product installed and available, you can encrypt the secured signon application key for added protection. In order to do so, use the KEYENCRYPTED keyword instead of KEYMASKED. Refer to *Security Server RACF Security Administrator’s Guide (SA22-7683)* for more information on this.

Attention: The client connection request will fail if PassTickets are not set up correctly.

Define z/OS UNIX program controlled files for RSE

Servers with authority to BPX.SERVER must run in a clean, program-controlled environment. This implies that all programs called by RSE must also be program controlled. For z/OS UNIX files, program control is managed by the **extattr** command. To execute this command, you need READ access to BPX.FILEATTR.PROGCTL in the FACILITY class, or be UID(0).

RSE server uses RACF's Java shared library (/usr/lib/libIRRRacf.so).

- extattr +p /usr/lib/libIRRRacf.so

Note:

- Since z/OS 1.9, /usr/lib/libIRRRacf.so is installed program controlled during SMP/E RACF install.
- Since z/OS 1.10, /usr/lib/libIRRRacf.so is part of SAF, which ships with base z/OS, so it is available also to non-RACF customers.
- The setup might be different if you use a security product other than RACF. Consult the documentation of your security product for more information.
- The SMP/E install of Developer for System z sets the program control bit for internal RSE programs.
- Use the **ls -Eog** z/OS UNIX command to display the current status of the program control bit (the file is program controlled if the letter **p** shows in the second string).

```
$ ls -Eog /usr/lib/libIRRRacf.so
-rwxr-xr-x  aps-  2      69632 Oct  5  2007 /usr/lib/libIRRRacf.so
```

Verify security settings

Use the following sample commands to display the results of your security-related customizations.

- Security settings and classes
 - SETROPTS LIST
- OMVS segment for users
 - LISTUSER #userid NORACF OMVS
 - LISTGRP #group-name NORACF OMVS
- Data set profiles
 - LISTGRP FEK
 - LISTDSD PREFIX(FEK) ALL
- Started tasks
 - LISTGRP STCGROUP OMVS
 - LISTUSER STCJMON OMVS
 - LISTUSER STCRSE OMVS
 - LISTUSER STCLOCK OMVS
 - RLIST STARTED JMON.* ALL STDATA
 - RLIST STARTED RSED.* ALL STDATA
 - RLIST STARTED LOCKD.* ALL STDATA
- JES command security

- RLIST CONSOLE JMON ALL
- RLIST OPERCMDS MVS.MCSOPER.JMON ALL
- RLIST OPERCMDS JES%.** ALL
- RSE as a secure z/OS UNIX server
 - RLIST FACILITY BPX.SERVER ALL
- MVS program controlled libraries for RSE
 - RLIST PROGRAM ** ALL
- Application protection for RSE
 - RLIST APPL FEKAPPL ALL
- PassTicket support for RSE
 - RLIST PTKTDATA FEKAPPL ALL SSIGNON
 - RLIST PTKTDATA IRRPTAUTH.FEKAPPL.* ALL
- z/OS UNIX program controlled files for RSE
 - ls -E /usr/lib/libIRRRacf.so

Chapter 11. Understanding Developer for System z

The Developer for System z host consists of several components that interact to give the client access to the host services and data. Understanding the design of these components can help you make the correct configuration decisions.

The following topics are covered in this chapter:

- “Component overview”
- “RSE as a Java application” on page 179
- “Task owners” on page 180
- “Connection flow” on page 182
- “Lock daemon” on page 183
- “z/OS UNIX directory structure” on page 185

Component overview

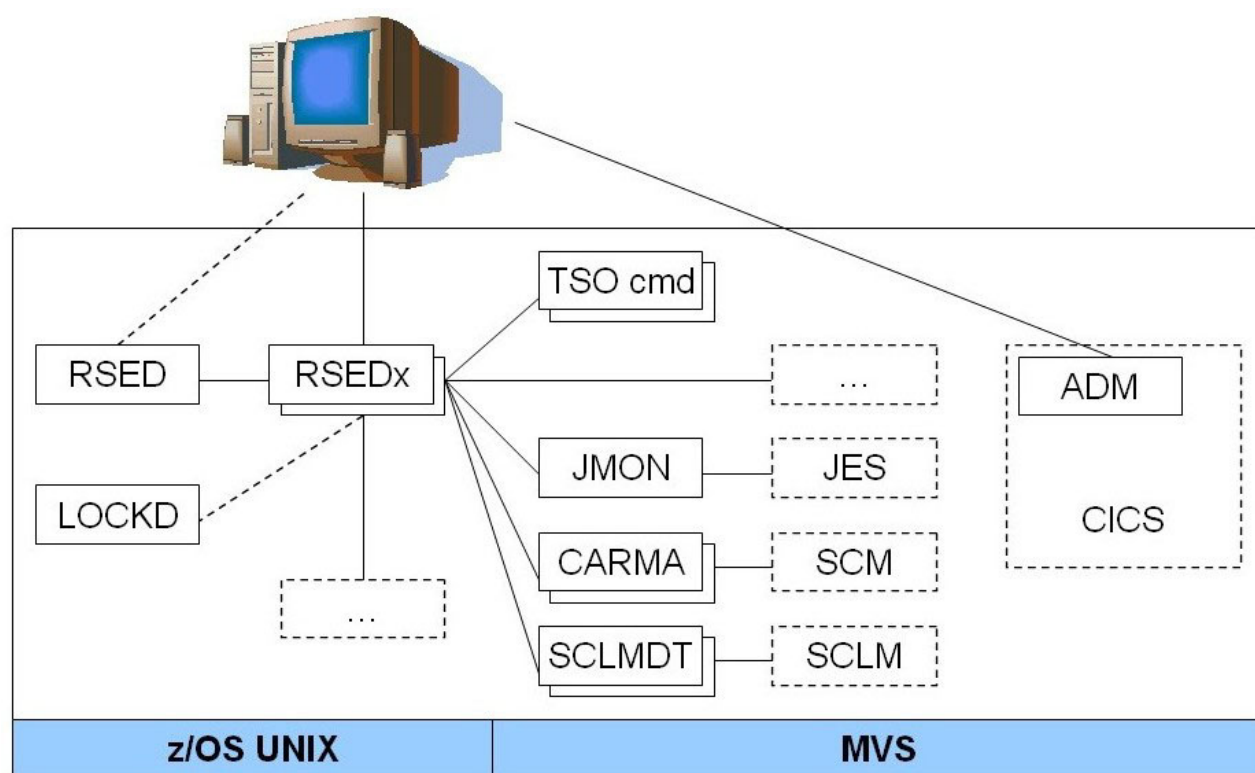


Figure 41. Component overview

Figure 41 shows a generalized overview of the Developer for System z layout on your host system.

- Remote Systems Explorer (RSE) provides core services, such as connecting the client to the host and starting other servers for specific services. RSE consists of two logical entities:

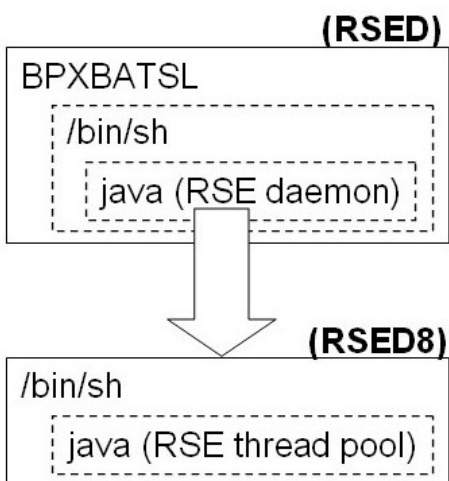
- RSE daemon (RSED), which manages connection setup. RSE daemon is also responsible for running in single server mode. To do so, RSE daemon creates one or more child processes known as RSE thread pools (RSEDx).
- RSE server, which handles individual client request. An RSE server is active as a thread inside a RSE thread pool.
- Lock Daemon (LOCKD) provides tracking services for data set locks.
- TSO Commands Service (TSO cmd) provides a batch-like interface for TSO and ISPF commands.
- JES Job Monitor (JMON) provides all JES related services.
- Common Access Repository Manager (CARMA) provides an interface to interact with Software Configuration Managers (SCMs), such as CA Endeavor.
- SCLM Developer Toolkit (SCLMDT) provides an interface to enhance and interact with SCLM.
- Application Deployment Manager (ADM) provides various CICS related services.
- More services are available, which can be provided by Developer for System z itself or corequisite software.

The description in the previous paragraph and list shows the central role assigned to RSE. With few exceptions, all client communication goes through RSE. This allows for easy security related network setup, as only a limited set of ports are used for client-host communication.

To manage the connections and workloads from the clients, RSE is composed of a daemon address space, which controls thread pooling address spaces. The daemon acts as a focal point for connection and management purposes, while the thread pools process the client workloads. Based upon the values defined in the `rsed.envvars` configuration file, and the amount of actual client connections, multiple thread pool address spaces can be started by the daemon.

RSE as a Java application

z/OS UNIX processes



Java storage usage

System - shared
System - private
Code (z/OS UNIX, Java, RSE)
Java heap
Not in use

JOBNAME	Status	PID	PPID	Command
RSED	FILE SYS KERNEL WAIT	50331904	1	BPXBATSL
RSED	WAITING FOR CHILD	67109114	50331904	/bin/sh ...
RSED	FILE SYS KERNEL WAIT	50331949	67109114	java ...
RSED8	WAITING FOR CHILD	307	50331949	/bin/sh ...
RSED8	FILE SYS KERNEL WAIT	308	307	java ...

Figure 42. RSE as a Java application

Figure 42 shows a basic view of resource usage (processes and storage) by RSE.

RSE is a Java application, which means that it is active in the z/OS UNIX environment. This allows for easy porting to different host platforms and straightforward communication with the Developer for System z client, which is also a Java application (based on the Eclipse framework). Therefore, basic knowledge of how z/OS UNIX and Java work is very helpful when you try to understand Developer for System z.

In z/OS UNIX, a program runs in a process, which is identified by a PID (Process ID). Each program is active in its own process, so invoking another program creates a new process. The process that started a process is referenced with a PPID (Parent PID), the new process is called a child process. The child process can run in the same address space or it can be spawned (created) in a new address space. A new process that runs in the same address space can be compared to executing a command in TSO, while the spawning one in a new address space is similar to submitting a batch job.

Note that a process can be single- or multi-threaded. In a multi-threaded application (such as RSE), the different threads compete for system resources as if they were separate address spaces (with less overhead).

Mapping this process information to the RSE sample in Figure 42, we get the following flow:

1. When the RSED task is started, it executes BPXBATSL, which invokes z/OS UNIX and creates a shell environment – PID 50331904.
2. In this process, the `rsed.sh` shell script is executed, which runs in a separate process (`/bin/sh`) – PID 67109114.
3. The shell script sets the environment variables defined in `rsed.envvars` and executes Java with the required parameters to start the RSE daemon – PID 50331949.
4. RSE daemon will spawn off a new shell in a child process (RSED8) – PID 307.
5. In this shell, the environment variables defined in `rsed.envvars` are set and Java is executed with the required parameters to start the RSE thread pool – PID 308.

Java applications, such as RSE, do not allocate storage directly, but use Java memory management services. These services, like allocating storage, freeing storage, and garbage collection, work within the limits of the Java heap. The minimum and maximum size of the heap is defined (implicitly or explicitly) during Java startup.

This implies that getting the most out of the available address space size is a balancing act of defining a large heap size while leaving enough room for z/OS to store a variable amount of system control blocks (dependant on the number of active threads).

Task owners

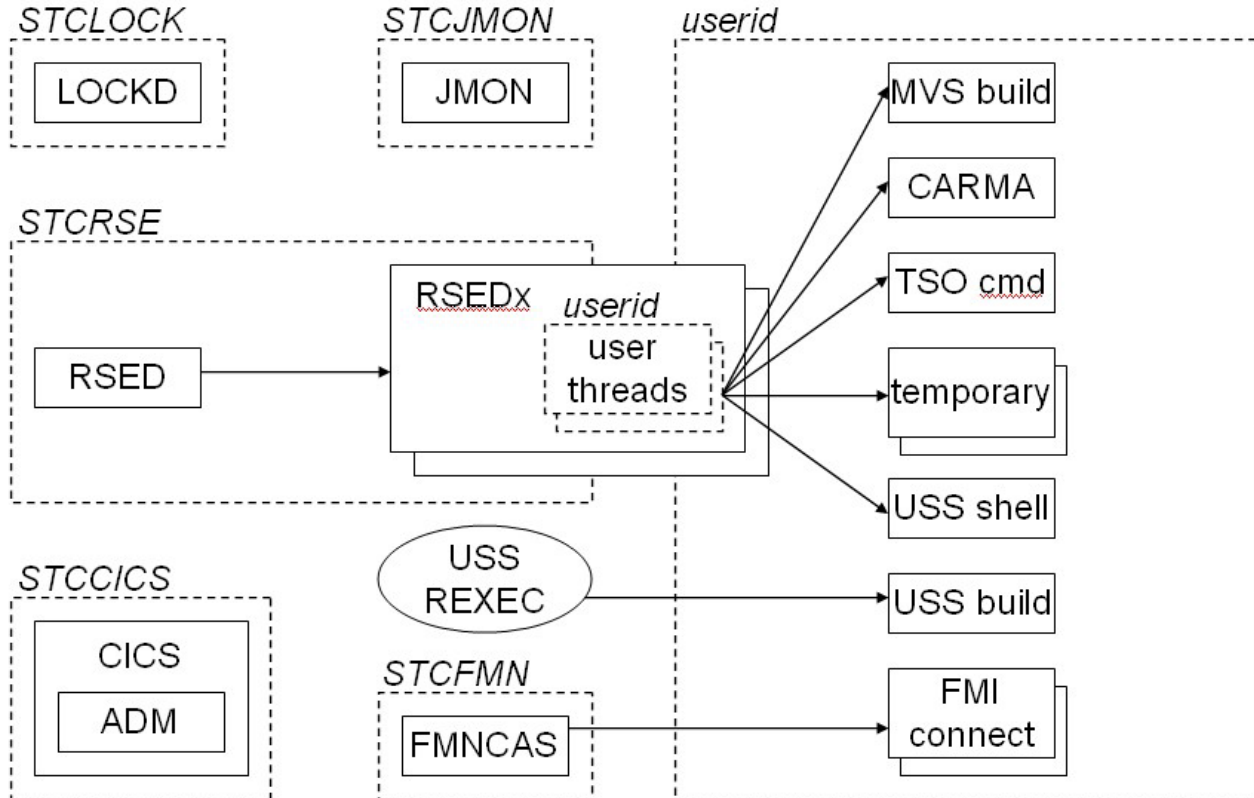


Figure 43. Task owners

Figure 43 on page 180 shows a basic overview of the owner of the security credentials used for various Developer for System z tasks.

The ownership of a task can be divided into two sections. Started tasks are owned by the user ID that is assigned to the started task in your security software. All other tasks, with the RSE thread pools (RSEDx) as exception, are owned by the client user ID.

Figure 43 on page 180 shows the Developer for system z started tasks (LOCKD, JMON and RSED), and sample started tasks and system services that Developer for System z communicates with. Application Deployment Manager (ADM) is active inside a CICS region. FMNCAS is the File Manager started task. The USS REXEC tag represents the z/OS UNIX REXEC (or SSH) service.

RSE daemon (RSED) creates one or more RSE thread pool address spaces (RSEDx) to process client requests. Each RSE thread pool supports multiple clients and is owned by the same user as the RSE daemon. Each client has his own threads inside a thread pool, and these threads are owned by the client user ID.

Depending on actions done by the client, one or more additional address spaces can be started, all owned by the client user ID, to perform the requested action. These address spaces can be an MVS batch job, an APPC transaction, or a z/OS UNIX child process. Note that a z/OS UNIX child process is active in a z/OS UNIX initiator (BPXAS), and the child process shows up as a started task in JES.

The creation of these address spaces is most often triggered by a user thread in a thread pool, either directly or by using system services like ISPF. But the address space could also be created by a third party. For example, File Manager will start a new address space for each data set (or member) it has to process on behalf of Developer for System z. z/OS UNIX REXEC or SSH are involved when starting builds in z/OS UNIX.

The user-specific address spaces end at task completion or when an inactivity timer expires. The started tasks remain active. The address spaces listed in Figure 43 on page 180 remain in the system long enough to be visible. However, you should be aware that due to the way z/OS UNIX is designed, there are also several short-lived temporary address spaces.

Connection flow

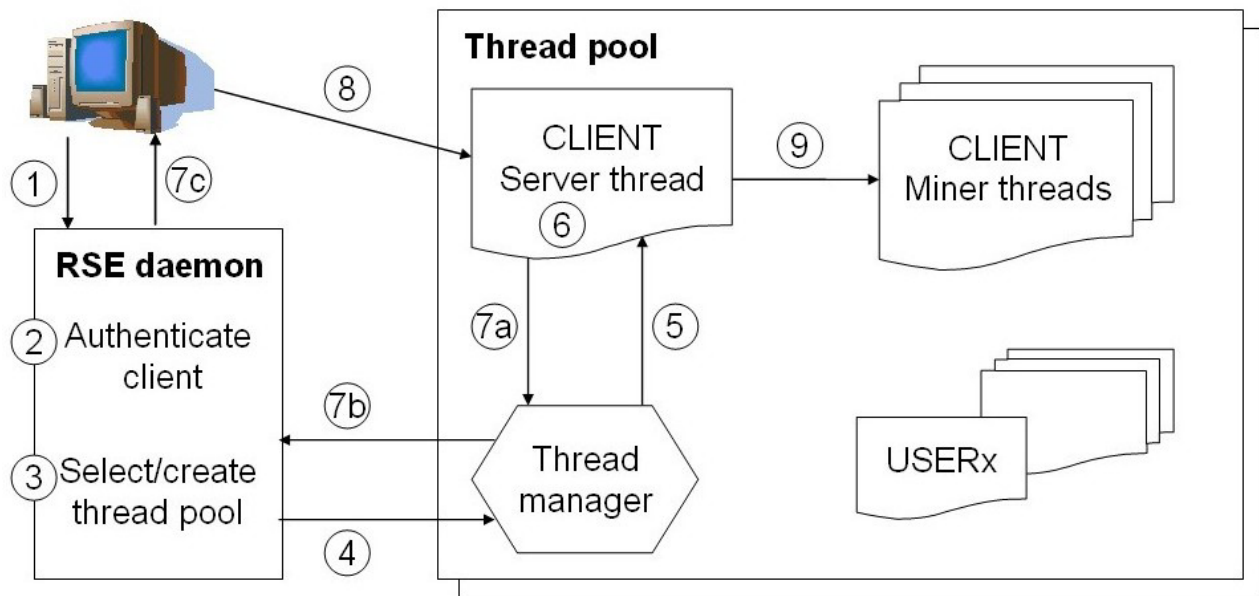


Figure 44. Connection flow

Figure 44 shows a schematic overview of how a client connects to the host using Developer for System z. It also briefly explains how PassTickets are used.

1. The client logs on to the daemon (port 4035).
2. RSE daemon authenticates the client, using the credentials presented by the client.
3. RSE daemon selects an existing thread pool or starts a new one if all are full.
4. RSE daemon passes the client user ID on to the thread pool.
5. The thread pool creates a client specific RSE server thread, using the client user ID and a PassTicket for authentication.
6. The client server thread binds to a port for future client communication.
7. The client server thread returns the port number for the client to connect to.
8. The client disconnects from RSE daemon and connects to the provided port number.
9. The client server thread starts other user specific threads (miners), always using the client user ID and a PassTicket for authentication. These threads provide the user-specific services requested by the client.

The description above shows the thread-oriented design of RSE. Instead of starting an address space per user, multiple users are serviced by a single thread pool address space. Within the thread pool, each miner (a user specific service) is active in its own thread with the user's security context assigned to it, ensuring a secure setup. This design accommodates large number of users with limited resource usage, but does imply that each client will use multiple threads (16 or more, depending on the performed tasks).

From a network point of view, Developer for system z acts similar to FTP in passive mode. The client connects to a focal point (RSE daemon) and then drops

the connection and reconnects to a port number provided by the focal point. The following logic controls the selection of the port that is used for the second connection:

1. If the client specified a non-zero port number in the subsystem properties tab, then RSE server will use that port for the bind. If this port is not available, the connection fails.
2. If `_RSE_PORTRANGE` is specified in `rsed.envvars`, then RSE server will bind to a port from this range. If no port is available, the connection fails. Note that RSE server does not need the port exclusively for the duration of the client connection. It is only in the time span between the (server) bind and the (client) connect that no other RSE server can bind to the port.
3. If no limitations are set, RSE server will bind to port 0. The result is that TCP/IP chooses the port number.

The usage of PassTickets for all z/OS services that require authentication allows Developer for System z to invoke these services at will without storing the password or constantly prompting the user for it. Use of PassTickets for all z/OS services also allows for alternative authentication methods during logon, such as one-time passwords and X.509 certificates.

Lock daemon

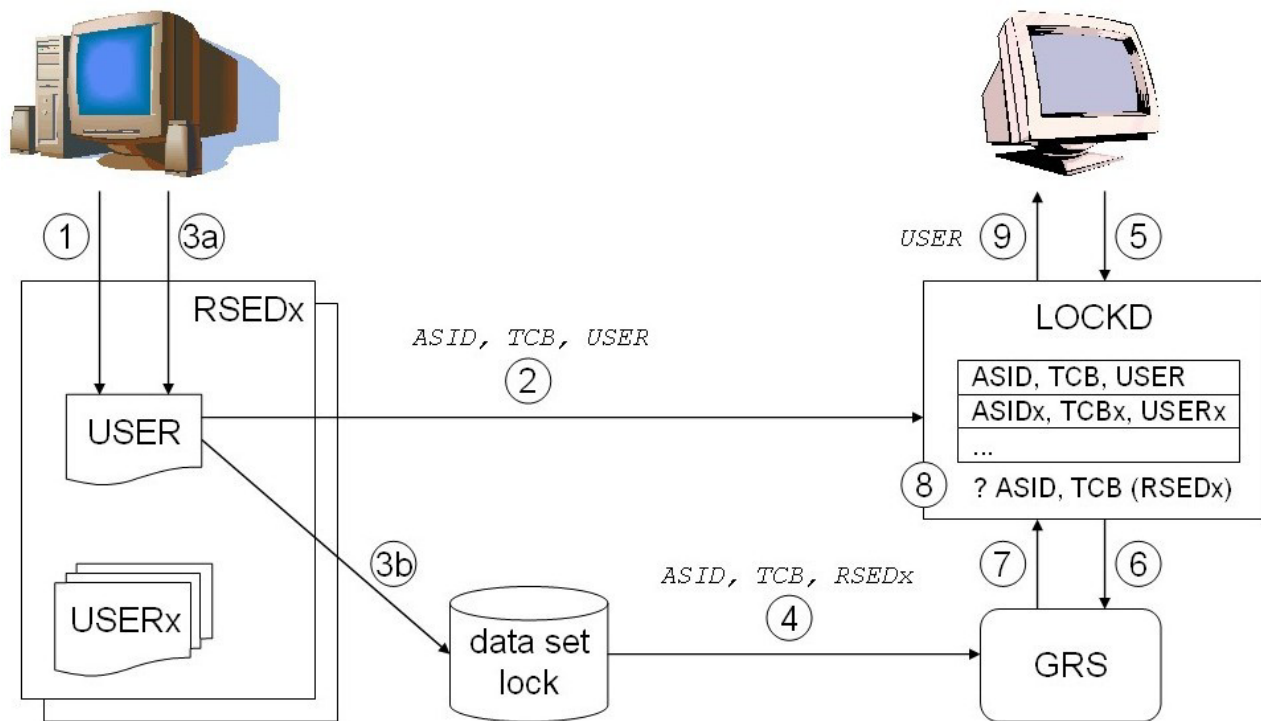


Figure 45. Lock daemon flow

Figure 45 shows a schematic overview of how the lock daemon determines which Developer for System z client owns a data set lock.

1. The client logs on, which creates a user-specific RSE server thread (USER) inside a thread pool (RSEDx).

2. RSE server registers a newly-connected user with the lock daemon. The registration information contains the Address Space Identifier (which is the ASID of the thread pool), the Task Control Block (TCB) identifier (user-specific), and the user ID.
3. The client opens a data set in edit, which instructs RSE server to get an exclusive lock on the data set.
4. The system registers the ASID, TCB and task name (RSEDx) of the requestor as part of lock process. This information is stored in the Global Resource Serialization (GRS) queues.
5. An operator (or RSE server on behalf of a client) queries the lock daemon for the lock status of the data set.
6. The lock daemon scans the GRS queues to learn if the data set is locked.
7. The daemon retrieves the ASID, TCB and task name of the lock owner.
8. The retrieved ASID and TCB are compared against the ASID and TCB combos of registered clients.
9. The related client user ID is returned to the requestor when a match is found. Otherwise, the task name retrieved from GRS is returned.

With the single-server setup of Developer for System z, where multiple users are assigned to a single thread pool address space, z/OS lost the ability to track who owns a lock on a data set or member. System commands stop at address space level, which is the thread pool.

To address this problem, Developer for System z provides the lock daemon. The lock daemon can track all dataset/member locks done by RSE users, as well as locks done by other products, such as ISPF.

RSE server registers a newly-connected user with the lock daemon. The registration information contains the Address Space Identifier (which is the ASID of the thread pool), the Task Control Block (TCB) ID (user-specific), and the user ID.

Note that registration is done at connect time only, so all RSE users active before the lock daemon was started (or restarted) will not be registered.

When the lock daemon receives a dataset query (by means of a modify query operator command or from the client by way of RSE server), the daemon scans the system's Global Resource Serialization (GRS) queues. If the ASID and TCB match that of a registered user, the user ID is returned as lock owner. Otherwise the jobname/user ID related to the ASID is returned as lock owner.

A console message (FEK513W) with the registration information is displayed if the registration fails. This allows an operator to match the values against the output of a **DISPLAY GRS,RES=(*,dataset*)** operator command in order to find the lock owner.

Note: Successful registrations are also listed in DD STDOUT of the server if log_level is set to 2. This is useful to do the manual mapping for successful registrations that were removed after a restart of the lock daemon.

Freeing a lock

Under normal circumstances, a data set or member is locked when the client opens it in edit mode, and freed when the client closes the edit session.

Certain error conditions can prevent this mechanism from working as designed. In this case, the user holding the lock can be canceled using RSE's **modify cancel** operator command, as described in Chapter 8, "Operator commands," on page 115. Active data set locks belonging to this user are freed during the process.

z/OS UNIX directory structure

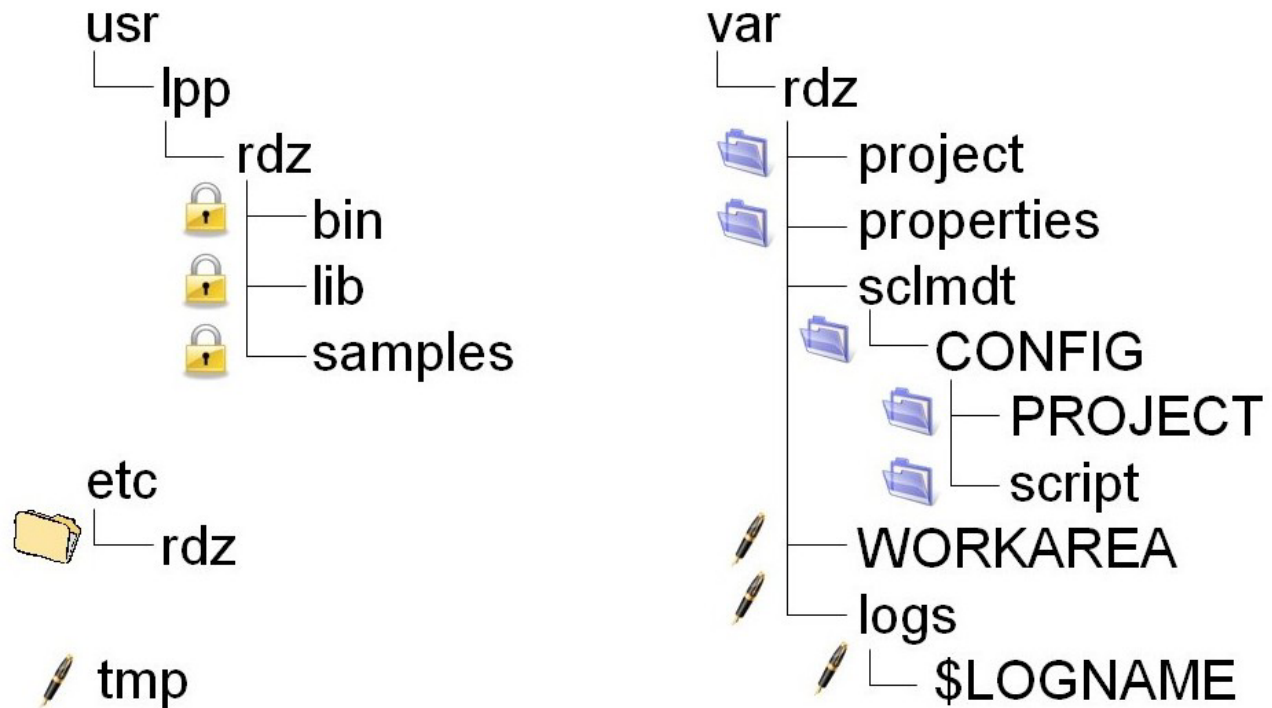


Figure 46. z/OS UNIX directory structure

Figure 46 shows an overview of the z/OS UNIX directories used by Developer for System z. The following list describes each directory touched by Developer for System z, how the location can be changed, and who maintains the data within.

- `/usr/lpp/rdz/` is the root path for the Developer for System z product code. The actual location is specified in the RSED and LOCKD started tasks (variable `HOME`). The files within are maintained by SMP/E.
- `/etc/rdz/` holds the RSE and miner related configuration files. The actual location is specified in the RSED and LOCKD started tasks (variable `CNFG`). The files within are maintained by the system programmer.
- `/var/rdz/sclmdt/CONFIG/` holds general SCLMDT configuration files. The actual location is specified in `rsed.envvars` (variable `SCLMDT_CONF_HOME`). The files within are maintained by the SCLM administrator.
- `/var/rdz/sclmdt/CONFIG/PROJECT/` holds SCLMDT project configuration files. The actual location is specified in `rsed.envvars` (variable `SCLMDT_CONF_HOME`). The files within are maintained by the SCLM administrator.
- `/var/rdz/sclmdt/CONFIG/script/` holds SCLMDT-related scripts that can be used by other products. The actual location is not specified anywhere. The files within are maintained by the SCLM administrator.
- `/var/rdz/projects/` holds the host-based project definition files. The actual location is specified in `projectcfg.properties` (variable `PROJECT-HOME`). The files within are maintained by a project manager or lead developer.

- `/var/rdz/properties/` holds the host-based property groups. The actual location is specified in `propertiescfg.properties` (variables `PROPERTY-GROUP` and `DEFAULT-VALUES`). The files within are maintained by a project manager or lead developer.
- `/var/rdz/logs/` holds the logs of RSE daemon and RSE thread pool servers. The actual location is specified in `rsed.envvars` (variable `daemon.log`). The files within are maintained by RSE.
- `/var/rdz/logs/$LOGNAME/` holds the user-specific logs of the RSE server and miners. The actual location is specified in `rsed.envvars` (variable `user.log` and `DSTORE_LOG_DIRECTORY`). The files within are maintained by RSE and the miners.

Note: `/var/rdz/logs/` requires permission bit mask 777 to allow each client to create his `$LOGNAME` directory and store the user-specific log files.

- `/var/rdz/WORKAREA/` is used by ISPF's TSO/ISPF Client Gateway and SCLMDT to transfer data between z/OS UNIX and MVS based address spaces. The actual location is specified in `rsed.envvars` (variable `_CMDSERV_WORK_HOME`). The files within are maintained by ISPF and SCLMDT.

Note: `/var/rdz/WORKAREA/` requires permission bit mask 777 to allow each client to create temporary files.

- `/tmp/` is used by ISPF's TSO/ISPF Client Gateway and various miners to store temporary data. The location is not customizable. The files within are maintained by ISPF and the miners. It is also the default location for Java dump files, which can be customized with the `_CEE_DUMPTARG` variable in `rsed.envvars`.

Note: `/tmp/` requires permission bit mask 777 to allow each client to create temporary files.

Update privileges for non-system administrators

The data in some directories, such as `/var/rdz/projects/`, is maintained by non-system administrators, such as project managers, who might not have many update privileges in z/OS UNIX. If there is just one user ID maintaining the files, there is not a problem after the user ID has been made owner of the directory and everything in the directory.

```
chown -R IBMUSER /var/rdz/projects/
```

When multiple user IDs need update permission to the directory, you can work with the group-permission bits.

1. Create a group in your security software that has a valid OMVS segment and connect all user ID's that require update access. This is preferably their default group.

```
ADDGROUP RDZPROJ OMVS(GID(1200))
CONNECT IBMUSER GROUP(RDZPROJ)
ALTUSER IBMUSER DFLTGRP(RDZPROJ)
```

2. Use the z/OS UNIX **chgrp** command to assign the group to the directory and all files within. This command must be repeated each time a new file is added and the desired group ID is not the default group for the user ID who added the file.

```
chgrp -R IBMUSER /var/rdz/projects/
```

3. Use the z/OS UNIX **chmod** command to give the whole group update permission to the directory and all files within.

```
chmod -R 775 /var/rdz/projects/
```

Chapter 12. WLM considerations

Unlike traditional z/OS applications, Developer for System z is not a monolithic application that can be identified easily to Workload Manager (WLM). Developer for System z consists of several components that interact to give the client access to the host services and data. As described in Chapter 11, “Understanding Developer for System z,” on page 177, some of these services are active in different address spaces, resulting in different WLM classifications.

The following topics are covered in this chapter:

- “Workload classification”
- “Setting goals” on page 189

Workload classification

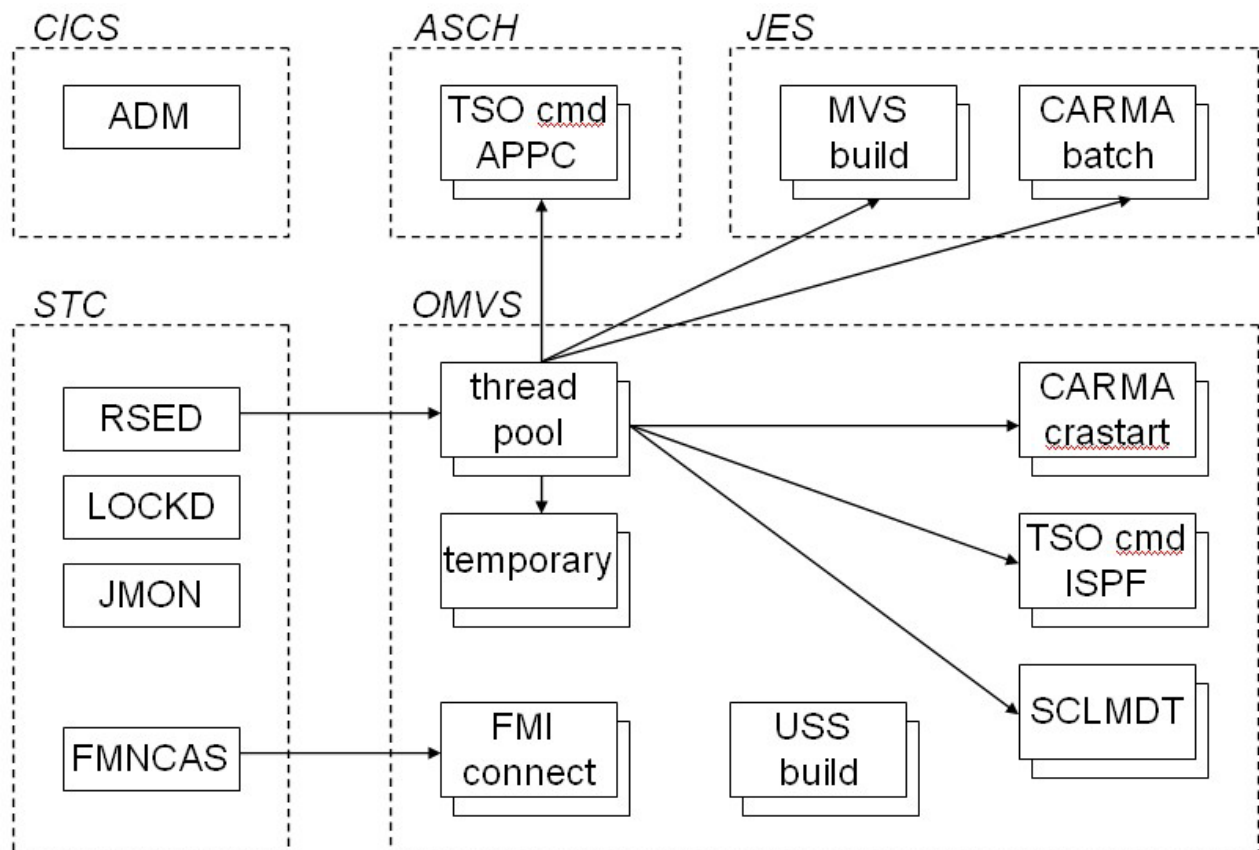


Figure 47. WLM classification

Figure 47 shows a basic overview of the subsystems via which Developer for System z workloads are presented to WLM.

Application Deployment Manager (ADM) is active within a CICS region, and will therefore follow the CICS classification rules in WLM.

RSE daemon (RSED), Lock daemon (LOCKD) and JES Job Monitor (JMON) are Developer for System z started tasks (or long-running batch jobs), each with their individual address space.

As documented in “RSE as a Java application” on page 179, RSE daemon spawns a child process for each RSE thread pool server (which supports a variable number of clients). Each thread pool is active in a separate address space (using a z/OS UNIX initiator, BPXAS). Because these are spawned processes, they are classified using the WLM OMVS classification rules, not the started task classification rules.

The clients that are active in a thread pool can create a multitude of other address spaces, depending on the actions done by the users. Depending on the configuration of Developer for System z, some workloads, such as the TSO Commands service (TSO cmd) or CARMA, can run in different subsystems.

The address spaces listed in Figure 47 on page 187 remain in the system long enough to be visible, but you should be aware that due to the way z/OS UNIX is designed, there are also several short-lived temporary address spaces. These temporary address spaces are active in the OMVS subsystem.

Note that while the RSE thread pools use the same user ID and a similar job name as the RSE daemon, all address spaces started by a thread pool are owned by the user ID of the client requesting the action. The client user ID is also used as (part of) the job name for all OMVS based address spaces started by the thread pool.

More address spaces are created by other services that Developer for System z uses, such as File Manager (FMNCAS) or z/OS UNIX REXEC (USS build).

Classification rules

WLM uses classification rules to map work coming into the system to a service class. This classification is based upon work qualifiers. The first (mandatory) qualifier is the subsystem type that receives the work request. Table 29 lists the subsystem types that can receive Developer for System z workloads.

Table 29. WLM entry-point subsystems

Subsystem type	Work description
ASCH	The work requests include all APPC transaction programs scheduled by the IBM-supplied APPC/MVS transaction scheduler, ASCH.
CICS	The work requests include all transactions processed by CICS.
JES	The work requests include all jobs that JES2 or JES3 initiates.
OMVS	The work requests include work processed in z/OS UNIX System Services forked children address spaces.
STC	The work requests include all work initiated by the START and MOUNT commands. STC also includes system component address spaces.

Table 30Table 2 lists additional qualifiers that can be used to assign a workload to a specific service class. Refer to MVS Planning: Workload Management (SA22-7602) for more details on the listed work qualifiers.

Table 30. WLM work qualifiers

		ASCH	CICS	JES	OMVS	STC
AI	Accounting Information	x		x	x	x

Table 30. WLM work qualifiers (continued)

		ASCH	CICS	JES	OMVS	STC
LU	LU Name (*)		x			
PF	Perform (*)			x		x
PRI	Priority			x		
SE	Scheduling Environment Name			x		
SSC	Subsystem Collection Name			x		
SI	Subsystem Instance (*)		x	x		
SPM	Subsystem Parameter					x
PX	Sysplex Name	x	x	x	x	x
SY	System Name (*)	x			x	x
TC	Transaction/Job Class (*)	x		x		
TN	Transaction/Job Name (*)	x	x	x	x	x
UI	User ID (*)	x	x	x	x	x

Note: For the qualifiers marked with (*), you can specify classification groups by adding a G to the type abbreviation. For example, a transaction name group would be TNG.

Setting goals

As documented in “Workload classification” on page 187, Developer for System z creates different types of workloads on your system. These different tasks communicate with each other, which implies that the actual elapse time becomes important to avoid time-out issues for the connections between the tasks. As a result, Developer for System z tasks should be placed in high-performance service classes, or in moderate-performance service classes with a high priority.

A revision, and possibly an update, of your current WLM goals is therefore advised. This is especially true for traditional MVS shops new to time-critical OMVS workloads.

Note:

- The goal information in this section is deliberately kept at a descriptive level, because actual performance goals are very site-specific.
- To help understand the impact of a specific task on your system, terms like minimal, moderate and substantial resource usage are used. These are all relative to the total resource usage of Developer for System z itself, not the whole system.

Table 31 lists the address spaces that are used by Developer for System z. z/OS UNIX will substitute "x" in the "Task Name" column by a random 1-digit number.

Table 31. WLM workloads

Description	Task name	Workload
JES Job Monitor	JMON	STC
Lock daemon	LOCKD	STC
RSE daemon	RSED	STC

Table 31. WLM workloads (continued)

Description	Task name	Workload
RSE thread pool	RSEDx	OMVS
ISPF Client Gateway (TSO Commands service and SCLMDT)	<userid>x	OMVS
TSO Commands service (APPC)	FEKFRSRV	ASCH
CARMA (batch)	CRA<port>	JES
CARMA (crastart)	<userid>x	OMVS
CARMA (ISPF Client Gateway)	<userid> and <userid>x	OMVS
MVS build (batch job)	*	JES
z/OS UNIX build (shell commands)	<userid>x	OMVS
z/OS UNIX shell	<userid>	OMVS
File Manager task	<userid>x	OMVS
Application Deployment Manager	CICSTS	CICS

Considerations for goal selection

The following general WLM considerations can help you to properly define the correct goal definitions for Developer for System z:

- You should base goals on what can actually be achieved, not what you want to happen. If you set goals higher than necessary, WLM moves resources from lower importance work to higher importance work which might not actually need the resources.
- Limit the amount of work assigned to the SYSTEM and SYSSTC service classes, because these classes have a higher dispatching priority than any WLM managed class. Use these classes for work that is of high importance but uses little CPU.
- Work that falls through the classification rules ends up in the SYSOTHER class, which has a discretionary goal. A discretionary goal tells WLM to just do the best it can when the system has spare resources.

When using response time goals:

- There must be a steady arrival rate of tasks (at least 10 tasks in 20 minutes) for WLM to properly manage a response time goal.
- Use average response time goals only for well controlled workloads, because a single long transaction has a big impact on the average response time and can make WLM overreact.

When using velocity goals:

- You usually cannot achieve a velocity goal above 90% for various reasons. For example, all the SYSTEM and SYSSTC address spaces have a higher dispatching priority than any velocity-type goal.
- WLM uses a minimum number of (using and delay) samples on which to base its velocity goal decisions. So the less work running in a service class, the longer it will take to collect the required number of samples and adjust the dispatching policy.
- Reevaluate velocity goals when you change your hardware. In particular, moving to fewer, faster processors requires changes to velocity goals.

STC

All Developer for System z started tasks, RSE daemon, Lock daemon and JES Job Monitor, are servicing real-time client requests.

Table 32. WLM workloads - STC

Description	Task name	Workload
JES Job Monitor	JMON	STC
Lock daemon	LOCKD	STC
RSE daemon	RSED	STC

- JES Job Monitor

JES Job Monitor provides all JES related services such as submitting jobs, browsing spool files and executing JES operator commands. You should specify a high-performance, one-period velocity goal, because the task does not report individual transactions to WLM. Resource usage depends heavily on user actions, and will therefore fluctuate, but is expected to be minimal to moderate.

- Lock daemon

The lock daemon queries the GRS enqueue tables upon client and operator request, and matches the result against known Developer for System z users. You should specify a high-performance, one-period velocity goal, because the task does not report individual transactions to WLM. Resource usage is expected to be minimal.

- RSE daemon

RSE daemon handles client logon and authentication, and manages the different RSE thread pools. You should specify a high-performance, one-period velocity goal, because the task does not report individual transactions to WLM. Resource usage is expected to be moderate, with a peak at the beginning of the workday.

OMVS

The OMVS workloads can be divided into two groups, RSE thread pools and everything else. This because all workloads, except RSE thread pools, use the client user ID as base for the address space name. (z/OS UNIX will substitute "x" in the "Task Name" column by a random 1-digit number.)

Table 33. WLM workloads - OMVS

Description	Task name	Workload
RSE thread pool	RSEDx	OMVS
ISPF Client Gateway (TSO Commands service and SCLMDT)	<userid>x	OMVS
CARMA (crastart)	<userid>x	OMVS
CARMA (ISPF Client Gateway)	<userid> and <userid>x	OMVS
z/OS UNIX build (shell commands)	<userid>x	OMVS
z/OS UNIX shell	<userid>	OMVS
File Manager task	<userid>x	OMVS

- RSE thread pool

An RSE thread pool is like the heart and brain of Developer for System z. Almost all data flows through here, and the miners (user specific threads) inside the thread pool control the actions of most other Developer for System z related tasks. You should specify a high-performance, one-period velocity goal, because the task does not report individual transactions to WLM. Resource usage depends heavily on user actions, and will therefore fluctuate, but is expected to be substantial.

The remaining workloads will all end up in the same service class due to a common address space naming convention. You should specify a multi-period goal for this service class. The first periods should be high-performance, percentile response time goals, while the last period should have a moderate-performance velocity goal. Some workloads, such as the ISPF Client Gateway, will report individual transactions to WLM, while others do not.

- ISPF Client Gateway

The ISPF Client Gateway is an ISPF service invoked by Developer for System z to execute non-interactive TSO and ISPF commands. This includes explicit commands issued by the client as well as implicit commands issued by Developer for System z, such as getting a PDS member list. Resource usage depends heavily on user actions, and will therefore fluctuate, but is expected to be minimal.

- CARMA

CARMA is an optional Developer for System z server that is used to interact with host based Software Configuration Managers (SCMs), such as CA Endevor[®] SCM. Developer for System z allows for different startup methods for a CARMA server, some of which become an OMVS workload. Resource usage depends heavily on user actions, and will therefore fluctuate, but is expected to be minimal.

- z/OS UNIX build

When a client initiates a build for a z/OS UNIX project, z/OS UNIX REXEC (or SSH) will start a task that executes a number of z/OS UNIX shell commands to perform the build. Resource usage depends heavily on user actions, and will therefore fluctuate, but is expected to be moderate to substantial, depending on the size of the project.

- z/OS UNIX shell

This workload processes z/OS UNIX shell commands that are issued by the client. Resource usage depends heavily on user actions, and will therefore fluctuate, but is expected to be minimal.

- IBM File Manager

Although not Developer for System z address spaces, the spawned File Manager child processes are listed here because they can be started upon request of a Developer for System z client, and these tasks use the same naming convention as Developer for System z tasks. These File Manager tasks process non-trivial MVS data set actions, such as formatted editing of a VSAM file. Resource usage depends heavily on user actions, and will therefore fluctuate, but is expected to be minimal to moderate.

JES

JES-managed batch processes are used in various manners by Developer for System z. The most common usage is for MVS builds, where a job is submitted and monitored to determine when it ends. But Developer for System z could also start a CARMA server in batch, and communicate with it using TCP/IP.

Table 34. WLM workloads - JES

Description	Task name	Workload
CARMA (batch)	CRA<port>	JES
MVS build (batch job)	*	JES

- CARMA

CARMA is an optional Developer for System z server that is used to interact with host based Software Configuration Managers (SCMs), such as CA Endeavor[®] SCM. Developer for System z allows for different startup methods for a CARMA server, some of which become a JES workload. You should specify a high-performance, one-period velocity goal, because the task does not report individual transactions to WLM. Resource usage depends heavily on user actions, and will therefore fluctuate, but is expected to be minimal.

- MVS build

When a client initiates a build for an MVS project, Developer for System z will start a batch job to perform the build. Resource usage depends heavily on user actions, and will therefore fluctuate, but is expected to be moderate to substantial, depending on the size of the project. Different moderate-performance goal strategies can be advisable, depending on your local circumstances.

- You could specify a multi-period goal with a percentile response time period and a trailing velocity period. In this case, your developers should be using mostly the same build procedure and similar sized input files to create jobs with uniform response times. There must also be a steady arrival rate of jobs (at least 10 jobs in 20 minutes) for WLM to properly manage a response time goal.
- A velocity goal is best suited for most batch-jobs, because this goal can handle highly variable execution times and arrival rates.

ASCH

In the current Developer for System z versions, the ISPF Client Gateway is used to execute non-interactive TSO and ISPF commands. Due to historical reasons, Developer for System z also supports executing these commands via an APPC transaction.

Table 35. WLM workloads - ASCH

Description	Task name	Workload
TSO Commands service (APPC)	FEKFRSRV	ASCH

- TSO Commands service

The TSO Commands service can be started as an APPC transaction by Developer for System z to execute non-interactive TSO and ISPF commands. This includes explicit commands issued by the client as well as implicit commands issued by Developer for System z, such as getting a PDS member list. You should specify a multi-period goal for this service class. For the first periods, you should specify high-performance, percentile response time goals. For the last period, you should specify a moderate-performance velocity goal. Resource usage depends heavily on user actions, and will therefore fluctuate, but is expected to be minimal.

CICS

Application Deployment Manager is an optional Developer for System z server that is active inside a CICS Transaction Server region.

Table 36. WLM workloads - CICS

Description	Task name	Workload
Application Deployment Manager	CICSTS	CICS

- **Application Deployment Manager**

The optional Application Deployment Manager server, which is active inside a CICSTS region, allows you to securely offload selected CICSTS management tasks to developers. Resource usage depends heavily on user actions, and will therefore fluctuate, but is expected to be minimal. The type of service class you should use depends on the other transactions active in this CICS region, and is therefore not discussed in detail.

WLM supports multiple types of management that you can use for CICS:

- **Managing CICS toward a region goal**

The goal is set to a service class that manages the CICS address spaces. You can only use an execution velocity goal for this service class. WLM uses the JES or STC classification rules for the address spaces but does not use the CICS subsystem classification rules for transactions.

- **Managing CICS toward a transaction response time goal**

A response time goal can be set in a service class assigned to a single transaction or a group of transactions. WLM uses the JES or STC classification rules for the address spaces and the CICS subsystem classification rules for transactions.

Chapter 13. Tuning considerations

As explained in Chapter 11, “Understanding Developer for System z,” on page 177, RSE (Remote Systems Explorer) is the core of Developer for System z. To manage the connections and workloads from the clients, RSE is composed of a daemon address space, which controls thread pooling address spaces. The daemon acts as a focal point for connection and management purposes, while the thread pools process the client workloads.

This makes RSE a prime target for tuning the Developer for System z setup. However, maintaining hundreds of users, each using 16 or more threads, a certain amount of storage, and possibly 1 or more address spaces requires proper configuration of both Developer for System z and z/OS.

The following topics are covered in this chapter:

- “Resource usage”
- “Storage usage” on page 205
- “z/OS UNIX file system space usage” on page 210
- “Key resource definitions” on page 212
- “Various resource definitions” on page 216
- “Monitoring” on page 217
- “Sample setup” on page 220

Resource usage

Use the information in this section to estimate the normal and maximum resource usage by Developer for System z, so you can plan your system configuration accordingly.

When you use the numbers and formulas presented in this section to define the values for system limits, be aware that you are working with fairly accurate estimates. Leave enough margin when setting the system limits to allow resource usage by temporary and other tasks, or by users connecting multiple times to the host simultaneously. (For example, by way of RSE and TN3270).

Note:

- The information is limited in scope to services accessed through RSE that are provided by Developer for System z itself. For example, resource usage of TN3270 is not documented (not accessed through RSE), nor is the resource usage of the programs called during remote (host-based) builds of MVS or z/OS UNIX projects (not provided by Developer for System z).
- Adding third-party extensions to Developer for System z can increase the resource usage counters.
- All services have short-lived “housekeeping” tasks, which use resources during their execution, and which may run sequential or parallel to each other. The resources used by these tasks are not documented.
- Where useful, user-specific resource usage of requisite software, such as the ISPF Client Gateway, is documented.
- The numbers presented here can change without prior notification.

Overview

The following tables give an overview of the number of address spaces, processes, and threads used by Developer for System z. More details on the numbers presented here can be found in the next sections:

- “Address space count” on page 197
- “Process count” on page 200
- “Thread count” on page 202

Table 37 gives a general overview of the key resources used by the Developer for System z started tasks. These resources are allocated only once. They are shared among all Developer for System z clients.

Table 37. Common resource usage

Started task	Address spaces	Processes	Threads
JMON	1	1	3
LOCKD	1	3	10
RSED	1	3	11
RSEDx	(a)	2	10

Note: (a) There is at least 1 RSE thread pool address space active. Refer to “Address space count” on page 197 to determine the actual number of RSE thread pool address spaces.

Table 38 gives a general overview of the key resources used by requisite software. These resources are allocated for each Developer for System z client that invokes the related function.

Table 38. User-specific requisite resource usage

Requisite software	Address spaces	Processes	Threads
ISPF Client Gateway	1	2	4
APPC	1	1	2
File Manager	1	1	2

Table 39 gives a general overview of the key resources used by each Developer for System z client when executing the specified function. Non-numeric values, such as ISPF, are a reference to the corresponding value in Table 38.

Table 39. User-specific resource usage

User action	Address spaces	Processes	Threads		
	User ID	User ID	User ID	RSEDx	JMON
Logon	-	-	-	16	1
Timer for idle timeout	-	-	-	1	-
Expand PDS(E)	ISPF	ISPF	ISPF	-	-
Open data set	ISPF	ISPF	ISPF	-	-

Table 39. User-specific resource usage (continued)

User action	Address spaces	Processes	Threads		
	User ID	User ID	User ID	RSEDx	JMON
TSO command	ISPF	ISPF	ISPF	-	-
z/OS UNIX shell	1	1	1	6	-
MVS build	1	-	-	-	-
z/OS UNIX build	3	3	3	-	-
CARMA (batch)	1	1	2	1	-
CARMA (crastart)	1	1	2	4	-
CARMA (ispf)	4	4	7	5	-
SCLMDT	ISPF	ISPF	ISPF	-	-
File Manager Integration	ISPF + FM	ISPF + FM	ISPF + FM	-	-
Fault Analyzer Integration	-	-	-	-	-

Note: ISPF can be substituted by APPC, except for SCLM Developer Toolkit.

Address space count

Table 40 lists the address spaces that are used by Developer for System z, where “u” in the “Count” column indicates that the amount must be multiplied by the number of concurrently active users using the function. z/OS UNIX will substitute “x” in the “Task Name” column by a random 1-digit number.

Table 40. Address space count

Count	Description	Task name	Shared	Ends after
1	JES Job Monitor	JMON	Yes	Never
1	Lock daemon	LOCKD	Yes	Never
1	RSE daemon	RSED	Yes	Never
(a)	RSE thread pool	RSEDx	Yes	Never
1u	ISPF Client Gateway (TSO Commands service and SCLMDT)	<userid>x	No	15 minutes or user logoff
1u	TSO Commands service (APPC)	FEKFRSRV	No	60 minutes or user logoff
1u	CARMA (batch)	CRA<port>	No	7 minutes or user logoff
1u	CARMA (crastart)	<userid>x	No	7 minutes or user logoff
4u	CARMA (ispf)	(1)<userid> or (3)<userid>x	No	7 minutes or user logoff
(b)	Simultaneous ISPF Client Gateway usage by 1 user	<userid>x	No	Task completion
1u	MVS build (batch job)	*	No	Task completion

Table 40. Address space count (continued)

Count	Description	Task name	Shared	Ends after
3u	z/OS UNIX build (shell commands)	<userid>x	No	Task completion
1u	z/OS UNIX shell	<userid>	No	User logoff
(c)	File Manager	<userid>x	No	Task completion

Note:

- (a) There is at least one RSE thread pool address space active. The actual number depends on:
 - The `minimum.threadpool.process` directive in `rsed.envvars`. The default value is 1.
 - The number of users that can be serviced by one thread pool. The default settings aim for 60 users per thread pool.
 - The high-water mark of concurrently active users, because idle thread pools are not stopped automatically.
- (b) Developer for System z has multiple threads active per user. In the event that the ISPF Client Gateway address space has not finished serving the request of one thread when another thread sends a request, ISPF will start up a new Client Gateway to process the new request. This address space ends after task completion.
- (c) The File Manager listener starts an address space per object that must be manipulated, for example a VSAM. This address space stays active until Developer for System z signals that the object is no longer needed, for example by closing the VSAM.
- SCLMDT requires an ISPF Client Gateway address space. SCLMDT shares the address space with the TSO Commands service.
- Most MVS data set-related actions use the TSO Commands service, which can be active in the ISPF Client Gateway or an APPC transaction, respectively.

Use the formula in Figure 48 to estimate the maximum number of address spaces used by Developer for System z.

$$3 + A + N \cdot (x + y + z) + (2 + N \cdot 0.01)$$

Figure 48. Maximum number of address spaces

Where

- “3” equals the number of permanent active server address spaces.
- “A” represents the number of RSE thread pool address spaces.
- “N” represents the maximum number of concurrent users.
- “x” is one of the following values, depending on the selected configuration options.

X	SCLMDT	TSO by way of Client Gateway	TSO by way of APPC
1	No	No	Yes
1	No	Yes	No

X	SCLMDT	TSO by way of Client Gateway	TSO by way of APPC
1	Yes	Yes	No

- “y” is one of the following values, depending on the selected configuration options.

Y	
0	No CARMA
1	CARMA (batch)
1	CARMA (crastart)
4	CARMA (ispf)

- “z” is 0 by default, but can increase depending on user actions:
 - Add 1 when a MVS build is performed. These address spaces end when the related build task (a batch job) completes.
 - Add 3 when a z/OS UNIX build is performed. Note that the actual number may be higher, depending on the needs of the programs invoked. These address spaces end when the related build task completes.
 - Add 1 for each concurrent interaction with IBM File Manager. These address spaces end when the requested object is no longer needed.
- “2 + N*0.01” adds a buffer for temporary address spaces. The required buffer size might differ at your site.

Use the formula in Figure 49 to estimate the maximum number of address spaces used by a Developer for System z client (not counting the undocumented temporary address spaces).

$$x + y + z$$

Figure 49. Number of address spaces per client

Where

- “x” depends on the selected configuration options and is documented for the formula to calculate the maximum number of address spaces (Figure 48 on page 198).
- “y” depends on the selected configuration options and is documented for the formula to calculate the maximum number of address spaces (Figure 48 on page 198).
- “z” is 0 by default, but can increase depending on user actions, as documented for the formula to calculate the maximum number of address spaces (Figure 48 on page 198).

The definitions in Table 41 can limit the actual number of address spaces.

Table 41. Address space limits

Location	Limit	Affected resources
rsed.envvars	maximum.threadpool.process	Limits the number of RSE thread pools
IEASYMxx	MAXUSER	Limits the number of address spaces
ASCHPMxx	MAX	Limits the number of APPC initiators for TSO Commands service (APPC)

Process count

Table 42 lists the number of processes per address space that is used by Developer for System z. “u” In the “Address Spaces” column indicates that the amount must be multiplied by the number of concurrently active users using the function.

Table 42. Process count

Processes	Address spaces	Description	User ID
1	1	JES Job Monitor	STCJMON
3	1	Lock daemon	STCLOCK
3	1	RSE daemon	STCRSE
2	(a)	RSE thread pool	STCRSE
2	(b)	ISPF Client Gateway (TSO Commands service and SCLMDT)	<userid>
1	1u	TSO Commands service (APPC)	<userid>
1	1u	CARMA (batch)	<userid>
1	1u	CARMA (crastart)	<userid>
1	1u	CARMA (ispf)	<userid>
1	3u	z/OS UNIX build (shell commands)	<userid>
1	1u	z/OS UNIX shell	<userid>
1	(c)	File Manager	<userid>
(5)	(u)	SCLM Developer Toolkit	<userid>

Note:

- (a) There is at least 1 RSE thread pool address space active. Refer to “Address space count” on page 197 to determine the actual number of RSE thread pool address spaces.
- RSE daemon and all RSE thread pools use the same user ID.
- (b) In normal situations, and when using the default configuration options, there is 1 ISPF Client Gateway active per user. The actual number can vary, as described in “Address space count” on page 197.
- (c) The File Manager listener uses a process per object that must be manipulated, for example a VSAM. This process stays active until Developer for System z signals that the object is no longer needed, for example by closing the VSAM.
- SCLMDT requires an ISPF Client Gateway address space. SCLMDT shares the address space with the TSO Commands service.
- (u) SCLMDT processes run in the ISPF Client Gateway address space, and thus do not have a value for the address space count.
- SCLMDT processes are temporary and end at task completion, but multiple processes can be active simultaneously for a single user. Table 42 lists the maximum number of concurrent SCLMDT processes.
- Most MVS data set-related actions use the TSO Commands service, which can be active in the ISPF Client Gateway or an APPC transaction, respectively.
- A z/OS UNIX build uses three processes in total, each running in their own address space.

- All listed processes stay active until the related address space ends, unless noted otherwise.

Use the formula in Figure 50 to estimate the maximum number of processes used by Developer for System z.

$$7 + 2 * A + N * (x + y + z) + (10 + N * 0.05)$$

Figure 50. Maximum number of processes

Where

- "7" equals the number of processes used by permanent active server address spaces.
- "A" represents the number of RSE thread pool address spaces.
- "N" represents the maximum number of concurrent users.
- "x" is one of the following values, depending on the selected configuration options.

X	SCLMDT	TSO by way of Client Gateway	TSO by way of APPC
1	No	No	Yes
2	No	Yes	No
7	Yes	Yes	No

- "y" is one of the following values, depending on the selected configuration options.

Y	
0	No CARMA
1	CARMA (batch)
1	CARMA (crastart)
4	CARMA (ispf)

- "z" is 0 by default, but can increase depending on user actions:
 - Add 1 when a z/OS UNIX shell is opened. This process stays active until the user logs off.
 - Add 3 when a z/OS UNIX build is performed. Note that the actual number may be higher, depending on the needs of the programs invoked. These processes end when the related build task completes.
 - Add 1 for each concurrent interaction with IBM File Manager. These processes end when the requested object is no longer needed.
- "10 + N*0.05" adds a buffer for temporary processes. The required buffer size might differ at your site.

Use the formula in Figure 51 on page 202 to estimate the maximum number of processes used by a Developer for System z client (not counting the undocumented temporary processes).

$$(x + y + z) + 5*s$$

Figure 51. Number of processes per client

Where

- "x" depends on the selected configuration options and is documented for the formula to calculate the maximum number of processes (Figure 50 on page 201).
- "y" depends on the selected configuration options and is documented for the formula to calculate the maximum number of processes (Figure 50 on page 201).
- "z" is 0 by default, but can increase depending on user actions, as documented for the formula to calculate the maximum number of processes (Figure 50 on page 201).
- "s" is 1 when SCLM Developer Toolkit is used, or 0 otherwise.

The definitions in Table 43 can limit the actual number of processes.

Table 43. Process limits

Location	Limit	Affected resources
BPXPRMxx	MAXPROCSYS	Limits the total number of processes
BPXPRMxx	MAXPROCUSER	Limits the number of processes per z/OS UNIX UID

Note:

- RSE daemon and the RSE thread pools use the same user ID. Since RSE daemon starts a new thread pool whenever needed, the number of processes for this user ID can grow. So MAXPROCUSER must be set to accommodate this growth, which can be formulated as "3 + 2*A".
- The MAXPROCUSER limit is per unique z/OS UNIX user ID (UID). Multiply the estimated per-user process count by the number of concurrently active clients if your users share the same UID.

Thread count

Table 44 lists the number of threads used by selected Developer for System z functions. "u" In the "Threads" columns indicates that the amount must be multiplied by the number of concurrently active users using the function. The thread count is listed per process, as limits are set at this level.

- RSEDx: These threads are created in the RSE thread pool, which is shared by multiple clients. All threads ending up in the same thread pool must be added together to get the total count.
- Active: These threads are part of the process that actually does the requested function. Each process is a stand-alone unit, so there is no need to sum the thread counts, even if they are assigned to same user ID, unless noted otherwise.
- Bootstrap: Bootstrap processes are needed to start the actual process. Each has 1 thread, and there can be multiple consecutive bootstraps. There is no need to sum the thread counts.

Table 44. Thread count

Threads			User ID	Description
RSEDx	Active	Bootstrap		

Table 44. Thread count (continued)

Threads			User ID	Description
-	3 + 1u	-	STCJMON	JES Job Monitor
-	10	2	STCLOCK	Lock daemon
-	11	2	STCRSE	RSE daemon
10 (a) + 16u	-	1 (a)	STCRSE	RSE thread pool
-	4u (b)	1u (b)	<userid>	ISPF Client Gateway (TSO Commands service and SCLMDT)
-	2u	-	<userid>	TSO Commands service (APPC)
1u	2u	-	STCRSE and <userid>	CARMA (batch)
4u	2u	-	STCRSE and <userid>	CARMA (crastart)
5u	4u	3u	STCRSE and <userid>	CARMA (ispf)
-	1u (d)	2u	<userid>	z/OS UNIX build (shell commands)
6u	1u	-	STCRSE and <userid>	z/OS UNIX shell
-	2u (c)	-	<userid>	File Manager
-	(5)	-	<userid>	SCLM Developer Toolkit
1u	-	-	STCRSE	Timer for idle timeout

Note:

- (a) There is at least 1 RSE thread pool address space active. Refer to “Address space count” on page 197 to determine the actual number of RSE thread pool address spaces.
- (b) In normal situations, and when using the default configuration options, there is 1 ISPF Client Gateway active per user. The actual number can vary, as described in “Address space count” on page 197.
- (c) There is one user-specific process (with the listed thread count) per interaction with IBM File Manager. These processes end when the requested object is no longer needed.
- SCLMDT requires an ISPF Client Gateway address space. SCLMDT shares the address space with the TSO Commands service.
- Depending on the selected action, SCLMDT can use multiple single-thread processes that end at task completion. Table 44 on page 202 lists the maximum number of concurrent SCLMDT threads.
- Most MVS data set-related actions use the TSO Commands service, which can be active in the ISPF Client Gateway or an APPC transaction, respectively.

- (d) A z/OS UNIX build invokes different build utilities, which might be multi-threaded. Table 44 on page 202 lists the minimum number of concurrent z/OS UNIX build threads.
- All listed threads stay active until the related process ends, unless noted otherwise.

Use the formula in Figure 52 to estimate the maximum number of threads used by a RSE thread pool. Use the formula in Figure 53 to estimate the maximum number of threads used by JES Job Monitor.

$$9 + N*(16 + x + y + z) + (20 + N*0.1)$$

Figure 52. Maximum number of RSE thread pool threads

$$3 + N$$

Figure 53. Maximum number of JES Job Monitor threads

Where

- "N" represents the maximum number of concurrent users in this thread pool or JES Job Monitor. The default settings aim for 60 users per thread pool.
- "x" is one of the following values, depending on the selected configuration options.

X	SCLMDT	TSO by way of Client Gateway	TSO by way of APPC	Timeout
0	No	No	Yes	No
0	No	Yes	No	No
0	Yes	Yes	No	No
1	No	No	Yes	Yes
1	No	Yes	No	Yes
1	Yes	Yes	No	Yes

- "y" is one of the following values, depending on the selected configuration options.

Y	
0	No CARMA
1	CARMA (batch)
4	CARMA (crastart)
5	CARMA (ispf)

- "z" is 0 by default, but can increase depending on user actions:
 - Add 6 when a z/OS UNIX shell is opened. These threads stay active until the user logs off.
- "20 + N*0.1" adds a buffer for temporary threads. The required buffer size might differ at your site.

The definitions in Table 45 can limit the actual number of threads in a process, which is mostly of importance for the RSE thread pools.

Table 45. Thread limits

Location	Limit	Affected resources
BPXPRMxx	MAXTHREADS	Limits the number of threads in a process.
BPXPRMxx	MAXTHREADTASKS	Limits the number of MVS tasks in a process.
BPXPRMxx	MAXASSIZE	Limits the address space size, and thus the storage available for thread related control blocks.
rsed.envvars	Xmx	Sets the maximum Java heap size. This storage is reserved and thus no longer available for thread related control blocks.
rsed.envvars	maximum.clients	Limits the number of clients (and thus their threads) in an RSE thread pool.
rsed.envvars	maximum.threads	Limits the number of client threads in a RSE thread pool.
FEJJCNFG	MAX_THREADS	Limits the number of threads in JES Job Monitor.

Note: The value for `maximum.threads` in `rsed.envvars` must be lower than the value for `MAXTHREADS` and `MAXTHREADTASKS` in `BPXPRMxx`.

Storage usage

RSE is a Java application, which implies that storage (memory) usage planning for Developer for System z must take two storage allocation limits into consideration, Java heap size and Address Space size.

Java heap size limit

Java offers many services to ease coding efforts for Java applications. One of these services is storage management.

Java's storage management allocates large blocks of storage and uses these for storage requests by the application. This storage managed by Java is called the Java heap. Periodic garbage collection (defragmentation) reclaims unused space in the heap and reduces its size.

The maximum Java heap size is defined in `rsed.envvars` with the `Xmx` directive. If this directive is not specified, Java uses a default size of 64 MB.

Each RSE thread pool (which services the client actions) is a separate Java application, and thus has a personal Java heap. Note that all thread pools use the same `rsed.envvars` configuration file, and thus have the same Java heap size limit.

The thread pool's usage of the Java heap depends heavily on the actions done by the connected clients. Regular monitoring of the heap usage is required to set the optimal heap size limit. Use the **modify display process** operator command to monitor the Java heap usage by RSE thread pools.

Address space size limit

All z/OS applications, including Java applications, are active within an address space, and are thus bound by address space size limitations.

The desired address space size is specified during startup, for example with the REGION parameter in JCL. However, system settings can limit the actual address space size. Refer to “Address Space size” on page 141 to learn more about these limits.

- MAXASSIZE in SYS1.PARMLIB(BPXPRMxx)
- ASSIZEMAX in the OMVS segment of the user ID assigned to the started task
- system exits IEFUSI and IEALIMIT

RSE thread pools inherit the address space size limits from RSE daemon. The address space size must be sufficient to house the Java heap, Java itself, common storage areas, and all control blocks the system creates to support the thread pool activity, such as a TCB (Task Control Block) per thread. Note that some of this storage usage is below the 16 MB line.

You should monitor the actual address space size before changing any settings that influence it, like changing the size of the Java heap or the amount of users supported by a single thread pool. Use your regular system monitoring software to track the actual storage usage by Developer for system z. If you do not have a dedicated monitoring tool, then basic information can be gathered with tools like the SDSF DA view or TASID (an as-is system information tool available via the ISPF "Support and downloads" webpage).

Size estimate guidelines

As stated before, the actual storage usage by Developer for system z is heavily influenced by user activity. Some actions use a fixed amount of storage (for example, logon), while others are variable (for example, listing data sets with a specified high level qualifier).

- Use a 2 GB address space for RSE to allow room for the Java heap and all the system control blocks.
- The sample rsed.envvars setup aims for 60 users per thread pool.
 - maximum.clients=60
 - maximum.threads=1000 ($10+16*60 = 970$, so 1000 allows for 61 clients)
- The sample rsed.envvars setup lets the Java heap grow up to 256 MB. This allows for 60 clients using an average of 4 MB per client ($60*4 = 240$).

Note that RSE displays the current Java heap and address space size limit during startup in console message FEK004I.

Use either of the following scenarios if monitoring shows that the current Java heap size is insufficient for the actual workload:

- Increase the maximum Java heap size with the Xmx directive in rsed.envvars. Before doing so, ensure that there is room in the address space for the size increase.
- Decrease the maximum number of clients per thread pool with the maximum.clients directive in rsed.envvars. RSE will still support the same number of clients, but the clients will be distributed among more thread pools.

Sample storage usage analysis

The displays in the following figures show some sample resource usage numbers for a default Developer for system z setup with one modification. The maximum Java heap size is set to 10 MB, as a small maximum will result in a bigger

percentile usage and the heap size limits will be reached sooner.

Max Heap Size=10MB and private AS Size=1,959MB

startup

BPXM023I (STCRSE)
ProcessId(268) Memory Usage(7%) Clients(0)

Jobname	Cpu time	Storage	EXCP
JMON	0.01	2740	72
LOCKD	1.60	28.7M	14183
RSED	4.47	32.8M	15910
RSED8	1.15	27.4M	12612

logon 1

BPXM023I (STCRSE)
ProcessId(268) Memory Usage(13%) Clients(1)

Jobname	Cpu time	Storage	EXCP
JMON	0.01	2864	81
LOCKD	1.64	28.8M	14259
RSED	4.55	32.8M	15980
RSED8	3.72	55.9M	24128

logon 2

BPXM023I (STCRSE)
ProcessId(268) Memory Usage(23%) Clients(2)

Jobname	Cpu time	Storage	EXCP
JMON	0.02	2944	86
LOCKD	1.66	28.9M	14268
RSED	4.58	32.9M	16027
RSED8	4.20	57.8M	25205

logon 3

BPXM023I (STCRSE)
ProcessId(268) Memory Usage(37%) Clients(3)

Jobname	Cpu time	Storage	EXCP
JMON	0.02	3020	91
LOCKD	1.67	29.0M	14277
RSED	4.60	32.9M	16076
RSED8	4.51	59.6M	26327

logon 4

BPXM023I (STCRSE)
ProcessId(268) Memory Usage(41%) Clients(4)

Jobname	Cpu time	Storage	EXCP
JMON	0.02	3108	96
LOCKD	1.68	29.0M	14286
RSED	4.61	32.9M	16125
RSED8	4.77	62.3M	27404

Figure 54. Resource usage with 5 logons

logon 5

BPXM023I (STCRSE)
ProcessId(268) Memory Usage(41%) Clients(4)
ProcessId(33554706) Memory Usage(13%) Clients(1)

Jobname	Cpu time	Storage	EXCP
JMON	0.03	3184	101
LOCKD	1.69	29.1M	14295
RSED	4.64	32.9M	16229
RSED8	4.78	62.4M	27413
RSED9	4.60	56.6M	24065

Figure 55. Resource usage with 5 logons (continued)

Figure 54 on page 207 and Figure 55 show a scenario where 5 clients log on to an RSE daemon with a 10 MB Java heap.

- A thread pool (RSED8) is in a dormant state upon startup, using about 27 MB, of which 0.7 MB is in the Java heap (7% of 10 MB).
- The thread pool becomes active when the first client connects, using another 27 MB plus 2 MB for each client that connects.
- Part of this 2MB per connection will be in the Java heap, as the increase in heap usage shows.
- However, there is no real pattern in the heap usage, because it depends on Java mechanisms that estimate the required storage and allocate more than needed. Intermittent garbage collection frees up storage, making trends even harder to detect.
- Internal mechanisms that limit the number of connections per thread pool to ensure sufficient heap size for the active threads result in the fifth connection being created in a new thread pool (RSED9). These internal safety nets are normally not invoked when using a properly configured setup, because other limits would be reached first (most likely the `maximum.clients` one in `rsed.envvars`).

Max Heap Size=10MB and private AS Size=1,959MB

startup

BPXM023I (STCRSE)
ProcessId(212) Memory Usage(7%) Clients(0)

Jobname	Cpu time	Storage	EXCP
JMON	0.01	2736	71
LOCKD	1.73	30.5M	14179
RSED	4.35	32.9M	15117
RSED8	1.43	27.4M	12609

logon

BPXM023I (STCRSE)
ProcessId(212) Memory Usage(13%) Clients(1)

Jobname	Cpu time	Storage	EXCP
JMON	0.01	2864	80
LOCKD	1.76	30.6M	14255
RSED	4.48	33.0M	15187
RSED8	3.53	53.9M	24125

expand large MVS tree (195 data sets)

BPXM023I (STCRSE)
ProcessId(212) Memory Usage(13%) Clients(1)

Jobname	Cpu time	Storage	EXCP
JMON	0.01	2864	80
LOCKD	1.78	30.6M	14255
RSED	4.58	33.1M	16094
RSED8	4.28	56.1M	24740

expand small PDS (21 members)

BPXM023I (STCRSE)
ProcessId(212) Memory Usage(13%) Clients(1)

Jobname	Cpu time	Storage	EXCP
IBMUSER2	0.22	2644	870
JMON	0.01	2864	80
LOCKD	1.78	30.6M	14255
RSED	4.61	33.1M	16108
RSED8	4.40	56.2M	24937

open medium sized member (86 lines)

BPXM023I (STCRSE)
ProcessId(212) Memory Usage(13%) Clients(1)

Jobname	Cpu time	Storage	EXCP
IBMUSER2	0.22	2644	870
JMON	0.01	2864	80
RSED	4.61	33.1M	16108
RSED8	8.12	62.7M	27044

Figure 56. Resource usage while editing a PDS member

Figure 56 shows a scenario where 1 client logs on to an RSE daemon with a 10 MB Java heap and edits a PDS member.

- The catalog search that results in 195 data set names used about 2MB of storage, all due to system activity, because the Java heap usage does not increase.
- Opening a 21-member PDS uses hardly any memory in the thread pool, but the display shows that TSO Commands service was invoked. There is a new address space active (IBMUUSER2), which uses the region size assigned to this user ID in TSO. This address space stays active for a specified amount of time so it can be reused for future requests by the TSO Commands service.
- Opening a member shows similar numbers as expanding a high level qualifier. The Java heap usage stays the same, but there is a 6.5 MB storage increase due to system activity.

z/OS UNIX file system space usage

Most of the Developer for System z related data that is not written to a DD statement ends up in a z/OS UNIX file. The system programmer has control over which data is written and where it goes. However, there is no control over the amount of data written.

The data can be grouped in the following categories:

- Problem analysis (log and system dump files), for which many details are documented in Chapter 9, “Troubleshooting configuration problems,” on page 127
- Auditing, as documented in “Audit logging” on page 152
- Temporary data

As documented in Chapter 9, “Troubleshooting configuration problems,” on page 127, Developer for System z writes the RSE-related host logs to the following z/OS UNIX directories:

- /var/rdz/logs for the RSE started task logs
- /var/rdz/logs/\$LOGNAME for user logs

By default, only error and warning messages are written to the logs. So if all goes as planned, these directories should hold only empty or nearly-empty files (not counting audit logs).

You can enable logging of informational messages, preferably under direction of the IBM support center, which increases the size of log files noticeably.

```

startup

$ ls -l /var/rdz/logs
total 144
-rw-rw-rw- 1 STCRSE STCGRP 33642 Jul 10 12:10 rsedaemon.log
-rw-rw-rw- 1 STCRSE STCGRP 1442 Jul 10 12:10 rseserver.log

logon

$ ls -l /var/rdz/logs
total 144
drwxrwxrwx 3 IBMUSER SYS1 8192 Jul 10 12:11 IBMUSER
-rw-rw-rw- 1 STCRSE STCGRP 36655 Jul 10 12:11 rsedaemon.log
-rw-rw-rw- 1 STCRSE STCGRP 1893 Jul 10 12:11 rseserver.log
$ ls -l /var/rdz/logs/IBMUSER
total 160
-rw-rw-rw- 1 IBMUSER SYS1 3459 Jul 10 12:11 ffs.log
-rw-rw-rw- 1 IBMUSER SYS1 0 Jul 10 12:11 ffsget.log
-rw-rw-rw- 1 IBMUSER SYS1 0 Jul 10 12:11 ffsput.log
-rw-rw-rw- 1 IBMUSER SYS1 303 Jul 10 12:11 lock.log
-rw-rw-rw- 1 IBMUSER SYS1 126 Jul 10 12:11 rmt_classloader_cache.jar
-rw-rw-rw- 1 IBMUSER SYS1 7266 Jul 10 12:11 rsecomm.log
-rw-rw-rw- 1 IBMUSER SYS1 0 Jul 10 12:11 stderr.log
-rw-rw-rw- 1 IBMUSER SYS1 0 Jul 10 12:11 stdout.log

logoff

$ ls -l /var/rdz/logs
total 80
drwxrwxrwx 3 IBMUSER SYS1 8192 Jul 10 12:11 IBMUSER
-rw-rw-rw- 1 STCRSE STCGRP 36655 Jul 10 12:11 rsedaemon.log
-rw-rw-rw- 1 STCRSE STCGRP 2208 Jul 10 12:11 rseserver.log
$ ls -l /var/rdz/logs/IBMUSER
total 296
-rw-rw-rw- 1 IBMUSER SYS1 6393 Jul 10 12:11 ffs.log
-rw-rw-rw- 1 IBMUSER SYS1 0 Jul 10 12:11 ffsget.log
-rw-rw-rw- 1 IBMUSER SYS1 0 Jul 10 12:11 ffsput.log
-rw-rw-rw- 1 IBMUSER SYS1 609 Jul 10 12:11 lock.log
-rw-rw-rw- 1 IBMUSER SYS1 126 Jul 10 12:11 rmt_classloader_cache.jar
-rw-rw-rw- 1 IBMUSER SYS1 45157 Jul 10 12:11 rsecomm.log
-rw-rw-rw- 1 IBMUSER SYS1 0 Jul 10 12:11 stderr.log
-rw-rw-rw- 1 IBMUSER SYS1 176 Jul 10 12:11 stdout.log

stop

$ ls -l /var/rdz/logs
total 80
drwxrwxrwx 3 IBMUSER SYS1 8192 Jul 10 12:11 IBMUSER
-rw-rw-rw- 1 STCRSE STCGRP 36655 Jul 10 12:11 rsedaemon.log
-rw-rw-rw- 1 STCRSE STCGRP 2490 Jul 10 12:12 rseserver.log

```

Figure 57. z/OS UNIX file system space usage

Figure 57 shows the minimal z/OS UNIX file system space usage when using debug level 2 (informational messages).

- The started task logs use 34 KB after startup and grow slowly when users log on, log off, or operator commands are issued.
- A client log directory uses 11 KB after logon and grows at a steady pace when the user starts working (not shown in the sample).
- Logoff adds another 40 KB to the user logs, bringing them to 51 KB.

Except for audit logs, log files are overwritten on every restart (for the RSE started task) or logon (for a client), keeping the total size in check. The `keep.last.log` directive in `rsd.envvars` changes this slightly, as it can instruct RSE to keep a copy of the previous logs. Older copies are always removed.

A warning message is sent to the console when the file system holding the audit log files is running low on free space and auditing is active. This console message (FEK103E) is repeated regularly until the low space issue is resolved. Refer to “Console messages” on page 121 for a list of console messages generated by RSE.

The definitions in Table 46 control which data is written to the log directories, and where the directories are located.

Table 46. Log output directives

Location	Directive	Function
resecmm.properties	debug_level	Set the default log detail level.
rsed.envvars	keep.last.log	Keep a copy of the previous logs before startup/logon.
rsed.envvars	enable.audit.log	Keep an audit trace of client actions.
rsed.envvars	enable.standard.log	Write the stdout and stderr streams of the thread pool (or pools) to a log file.
rsed.envvars	DSTORE_TRACING_ON	Enable DataStore action logging.
rsed.envvars	DSTORE_MEMLOGGING_ON	Enable DataStore memory usage logging.
Operator command	modify rsecommlog <level>	Dynamically change the log detail level of rsecomm.log
Operator command	modify rsedaemonlog <level>	Dynamically change the log detail level of rsedaemon.log
Operator command	modify rseserverlog <level>	Dynamically change the log detail level of rseserver.log
Operator command	modify rsestandardlog {on off}	Dynamically change the updating of std*.log
rsed.envvars	daemon.log	Home path for RSE started task and audit logs.
rsed.envvars	user.log	Home path for user logs.

Developer for System z, together with requisite software such as the ISPF Client Gateway, also writes temporary data to /tmp and /var/rdz/WORKAREA. The amount of data written here as a result of user actions is unpredictable, so you should have ample free space in the file systems holding these directories.

Developer for system z always tries to clean up these temporary files, but manual cleanup, as documented in “(Optional) WORKAREA cleanup” on page 98, can be performed at virtually any time.

Key resource definitions

/etc/rdz/rsed.envvars

The environment variables defined in rsed.envvars are used by RSE, Java, and z/OS UNIX. The sample file that comes with Developer for System z is targeted at small to medium sized installations that do not require the optional components of Developer for System z. “rsed.envvars, RSE configuration file” on page 28 describes each variable that is defined in the sample file, where the following variables require special attention:

_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -Xms128m -Xmx256m"

Set initial (Xms) and maximum (Xmx) heap size. The defaults are 128M and 256M respectively. Change to enforce the desired heap size values. If this directive is commented out, the Java default values will be used, which are 4M and 512M respectively (1M and 64M for Java 5.0).

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -Dmaximum.clients=60"

Maximum amount of clients serviced by one thread pool. The default is 60. Uncomment and customize to limit the number of clients per thread pool. Note that other limits may prevent RSE from reaching this limit.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -Dmaximum.threads=1000"

Maximum amount of active threads in one thread pool to allow new clients. The default is 1000. Uncomment and customize to limit the number of clients per thread pool based upon the number of threads in use. Note that each client connection uses multiple threads (16 or more) and that other limits may prevent RSE from reaching this limit.

Note: This value must be lower than the setting for MAXTHREADS and MAXTHREADTASKS in SYS1.PARMLIB(BPXPRMxx).

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -Dminimum.threadpool.process=10"

The minimum number of active thread pools. The default is 1. Uncomment and customize to start at least the listed number of thread pool processes. Thread pool processes are used for load balancing the RSE server threads. More new processes are started when they are needed. Starting the new processes up front helps prevent connection delays but uses more resources during idle times.

#_RSE_JAVAOPTS="\$_RSE_JAVAOPTS -Dmaximum.threadpool.process=100"

The maximum number of active thread pools. The default is 100. Uncomment and customize to limit the number of thread pool processes. Thread pool processes are used for load balancing the RSE server threads, so limiting them will limit the amount of active client connections.

SYS1.PARMLIB(BPXPRMxx)

RSE is a Java application, which means that it is active in the z/OS UNIX environment. This promotes BPXPRMxx to become a crucial parmlib member, as it contains the parameters that control the z/OS UNIX environment and file systems. BPXPRMxx is described in the *MVS Initialization and Tuning Reference* (SA22-7592). The following directives are known to impact Developer for System z:

MAXPROCSYS(nnnnn)

Specifies the maximum number of processes that the system allows.

Value Range: nnnnn is a decimal value from 5 to 32767.

Default: 900

MAXPROCUSER(nnnnn)

Specifies the maximum number of processes that a single z/OS UNIX user ID can have concurrently active, regardless of how the processes were created.

Value Range: nnnnn is a decimal value from 3 to 32767.

Default: 25

Note:

- All RSE processes use the same z/OS UNIX user ID (that of the user who is assigned to RSE daemon), because all clients run as threads within the RSE processes.
- This value can also be set with the PROCUSERMAX variable in the OMVS security profile segment of the user assigned to the RSED started task.

MAXTHREADS(nnnnnn)

Specifies the maximum number of pthread_created threads, including running, queued, and exited but undetached, that a single process can have concurrently active. Specifying a value of 0 prevents applications from using pthread_create.

Value Range: nnnnnn is a decimal value from 0 to 100000.

Default: 200

Note:

- Each client uses at least 16 threads within the RSE thread pool process, and multiple clients are active within the process.
- This value can also be set with the THREADSMAX variable in the OMVS security profile segment of the user assigned to the RSED started task. When set, the THREADSMAX value is used for both MAXTHREADS and MAXTHREADTASKS.

MAXTHREADTASKS(nnnnnn)

Specifies the maximum number of MVS tasks that a single process can have concurrently active for pthread_created threads.

Value Range: nnnnnn is a decimal value from 0 to 32768.

Default: 1000

Note:

- Each active thread has an MVS task (TCB, Task Control Block).
- Each concurrent MVS task requires additional storage, some of which must be below the 16MB line.
- Each client uses at least 16 threads within the RSE thread pool process, and multiple clients are active within the process.
- This value can also be set with the THREADSMAX variable in the OMVS security profile segment of the user assigned to the RSED started task. When set, the THREADSMAX value is used for both MAXTHREADS and MAXTHREADTASKS.

MAXUIDS(nnnnnn)

Specifies the maximum number of z/OS UNIX user IDs (UIDs) that can operate concurrently.

Value Range: nnnnnn is a decimal value from 1 to 32767.

Default: 200

MAXASSIZE(nnnnnn)

Specifies the RLIMIT_AS resource values that will be established as the initial values for new processes. RLIMIT_AS indicates the address space region size.

Value Range: nnnnnn is a decimal value from 10485760 (10 Megabytes) to 2147483647 (2 Gigabytes).

Default: 209715200 (200 Megabytes)

Note:

- This value should be set to 2G.
- This value can also be set with the ASSIZEMAX variable in the OMVS security profile segment of the user assigned to the RSED started task.

MAXFILEPROC(nnnnnn)

Specifies the maximum number of descriptors for files, sockets, directories, and any other file system objects that a single process can have concurrently active or allocated.

Value Range: nnnnnn is a decimal value from 3 to 524287.

Default: 64000

Note:

- A thread pool has all his client threads in a single process.
- This value can also be set with the FILEPROCMAx variable in the OMVS security profile segment of the user assigned to the RSED started task.

MAXMMAPAREA(nnnnnn)

Specifies the maximum amount of data space storage space (in pages) that can be allocated for memory mappings of z/OS UNIX files. Storage is not allocated until the memory mapping is active.

Value Range: nnnnnn is a decimal value from 1 to 16777216.

Default: 40960

Note: This value can also be set with the MMAPAREAMAX variable in the OMVS security profile segment of the user assigned to the RSED started task.

Use the **SETOMVS** or **SET OMVS** operator command to dynamically (until next IPL) increase or decrease the value of any of the previous BPXPRMxx variables. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs. Refer to *MVS System Commands* (SA22-7627) for more information on these operator commands.

The following definitions are sub-parameters of the NETWORK statement.

MAXSOCKETS(nnnnnnnn)

Specifies the maximum number of sockets supported by this file system for this address family. This is an optional parameter.

Value Range: nnnnnnnn is a decimal value from 0 to 16777215.

Default: 100

INADDRANYCOUNT(nnnn)

Specifies the number of ports that the system reserves for use with PORT 0, INADDR_ANY binds, starting with the port number specified in the INADDRANYPORT parameter. This value is only needed for CINET (multiple TCP/IP stacks).

Value Range: nnnn is a decimal value from 1 to 4000.

Default: If neither INADDRANYPORT or INADDRANYCOUNT is specified, the default for INADDRANYCOUNT is 1000. Otherwise, no ports are reserved (0).

Various resource definitions

EXEC card in the server JCL

The following definitions are recommended to be added to the EXEC card in the JCL of the Developer for System z servers.

REGION=0M

REGION=0M is recommended for the RSE daemon and JES Job Monitor started tasks, RSED and JMON respectively. By doing so, the address space size is limited only by the available private storage, or by the IEFUSI or IEALIMIT system exits. Note that IBM strongly recommends not to use these exits for z/OS UNIX address spaces, like RSE daemon.

TIME=NOLIMIT

TIME=NOLIMIT is recommended to be used for all Developer for System z servers. This because the CPU time of all Developer for System z clients accumulates in the server address spaces.

FEK.#CUST.PARMLIB(FEJJCNFG)

The environment variables defined in FEJJCNFG are used by JES Job Monitor. The sample file that comes with Developer for System z is targeted at small to medium sized installations. “FEJJCNFG, JES Job Monitor configuration file” on page 24 describes each variable that is defined in the sample file, where the following variables require special attention:

MAX_THREADS

Maximum number of users that can be using one JES Job Monitor at a time. The default is 200. The maximum value is 2147483647. Increasing this number may require you to increase the size of the JES Job Monitor address space.

SYS1.PARMLIB(IEASYSxx)

IEASYSxx holds system parameters and is described in the *MVS Initialization and Tuning Reference* (SA22-7592). The following directives are known to impact Developer for System z:

MAXUSER=nnnnn

This parameter specifies a value that, under most conditions, the system uses to limit the number of jobs and started tasks that can run concurrently during a given IPL.

Value Range: nnnnn is a decimal value from 0-32767. Note that the sum of the values specified for the MAXUSER, RSVSTRT, and RSVNONR system parameters cannot exceed 32767.

Default Value: 255

SYS1.PARMLIB(IVTPRMxx)

IVTPRMxx sets parameters for the Communication Storage Manager (CSM), and is described in the *MVS Initialization and Tuning Reference* (SA22-7592). The following directives are known to impact Developer for System z:

FIXED MAX(maxfix)

Defines the maximum amount of storage dedicated to fixed CSM buffers.

Value Range: maxfix is a value from 1024K to 2048M.

Default: 100M

ECSA MAX(maxecsa)

Defines the maximum amount of storage dedicated to ECSA CSM buffers.

Value Range: maxecsa is a value from 1024K to 2048M.

Default: 100M

SYS1.PARMLIB(ASCHPMxx)

The ASCHPMxx parmlib member contains scheduling information for the ASCH transaction scheduler, and is described in the *MVS Initialization and Tuning Reference* (SA22-7592). The following directives are known to impact Developer for System z:

MAX(nnnnn)

An optional parameter of the CLASSADD definition that specifies the maximum number of APPC transaction initiators that are allowed for a particular class of transaction initiators. After this limit is reached, no new address spaces are created and incoming requests are queued to wait until existing initiator address spaces become available. The value should not exceed the maximum number of address spaces allowed by your installation, and you should be aware of competing products on the system that will also require address spaces.

Value Range: nnnnn is a decimal value from 1 to 64000.

Default: 1

Note: If you use APPC to start the TSO Commands service, then the transaction class used must have enough transaction initiators to allow one initiator for each concurrent user of Developer for System z.

Monitoring

Since user workloads can change the need for system resources, the system should be monitored regularly to measure resource usage so that Rational Developer for System z and system configurations can be adjusted in response to user requirements. The following commands can be used to aid in this monitoring process.

Monitoring RSE

RSE thread pools are the focal point for user activity in Developer for System z, and thus require monitoring for optimal use. RSE daemon can be queried for information that cannot be gathered with regular system monitoring tools.

- Use your regular system monitoring tools, such as RMF™, to gather address space-specific data such as used real storage and CPU-time. If you do not have a dedicated monitoring tool, then basic information can be gathered with tools like the SDSF DA view or TASID (an as-is system information tool available via the ISPF “Support and downloads” webpage).
- During startup, the RSE daemon reports the available address space size and Java heap size with console message FEK004I.
FEK004I RseDaemon: Max Heap Size=65MB and private AS Size=1,959MB
- The **MODIFY RSED,APPL=DISPLAY PROCESS** operator command displays the RSE thread pool processes. The “Memory Usage” field shows how much of the defined Java heap is actually used. Refer to Chapter 8, “Operator commands,” on page 115 for more information on this command.

```
f rsed,appl=d p
BPXM023I (STCRSE)
ProcessId(16777456) Memory Usage(33%) Clients(4) Order(1)
```

More information is provided when the DETAIL option of the **DISPLAY PROCESS** modify command is used:

```
f rsed,appl=d p,detail
BPXM023I (STCRSE)
ProcessId(33555087) ASId(002E) JobName(RSED8) Order(1)
PROCESS LIMITS:  CURRENT  HIGHWATER  LIMIT
  JAVA HEAP USAGE(%)  10      56      100
  CLIENTS              0      25      60
  MAXFILEPROC          83     103     64000
  MAXPROCUSER          97     99      200
  MAXTHREADS           9      14     1500
  MAXTHREADTASKS       9      14     1500
```

Monitoring z/OS UNIX

Most z/OS UNIX limits that are of interest for Developer for System z can be displayed using operator commands. Some commands even show the current usage and the high-water mark for a specific limit. Refer to *MVS System Commands* (SA22-7627) for more information on these commands.

- The **LIMMSG(ALL)** directive in **SYS1.PARMLIB(BPXPRMxx)** tells z/OS UNIX to display console messages (BPXI040I) when any of the parmlib limits is about to be reached. The default value for LIMMSG is NONE, which disables the function. Use operator command **SETOMVS LIMMSG=ALL** to dynamically activate this function (until next IPL). Refer to *MVS Initialization and Tuning Reference* (SA22-7592) for more information on this directive.
- The **DISPLAY OMVS,OPTIONS** operator command displays the current values of z/OS UNIX directives that can be set dynamically.

```
d omvs,o
BPX0043I 13.10.16 DISPLAY OMVS 066
OMVS 000D ETC/INIT WAIT OMVS=(M7)
CURRENT UNIX CONFIGURATION SETTINGS:
MAXPROCSYS = 256 MAXPROCUSER = 16
MAXFILEPROC = 256 MAXFILESIZE = NOLIMIT
MAXCPUPTIME = 1000 MAXUIDS = 200
MAXPTYs = 256
MAXMMAPAREA = 256 MAXASSIZE = 209715200
MAXTHREADS = 200 MAXTHREADTASKS = 1000
MAXCORESIZE = 4194304 MAXSHAREPAGES = 4096
IPCMSGQBYTES = 2147483647 IPCMSGQMNUM = 10000
IPCMSGNIDS = 500 IPCSEMNIIDS = 500
IPCSEMNIOPS = 25 IPCSEMNISEMS = 1000
IPCshmMPAGES = 25600 IPCshmNIIDS = 500
IPCshmNSEGS = 500 IPCshmSPAGES = 262144
SUPERUSER = BPXROOT FORKCOPY = COW
STEPLIBLIST =
USERIDALIASTABLE=
SERV_LINKLIB = POSIX.DYNSERV.LOADLIB BPXLK1
SERV_LPALIB = POSIX.DYNSERV.LOADLIB BPXLK1
PRIORITYPG VALUES: NONE
PRIORITYGOAL VALUES: NONE
MAXQUEUEDSIGs = 1000 SHRLIBRGNSIZE = 67108864
SHRLIBMAXPAGES = 4096 VERSION = /
SYSCALL COUNTS = NO TTYGROUP = TTY
SYSPLX = NO BRLM SERVER = N/A
LIMMSG = NONE AUTOCVT = OFF
RESOLVER PROC = DEFAULT
AUTHPGMLIST = NONE
SWA = BELOW
```

- The **DISPLAY OMVS,LIMITS** operator command displays information about current z/OS UNIX System Services parmlib limits, their high-water marks, and current system usage.

```

d omvs,l
BPX0051I 14.05.52 DISPLAY OMVS 904
OMVS      0042 ACTIVE          OMVS=(69)
SYSTEM WIDE LIMITS:          LIMMSG=SYSTEM
                                CURRENT  HIGHWATER  SYSTEM
                                USAGE    USAGE    LIMIT
MAXPROCSYS                   1           4       256
MAXUIDS                      0           0       200
MAXPTYS                        0           0       256
MAXMMAPAREA                  0           0       256
MAXSHAREPAGES                  0           0       4096
IPCMSGNIDS                     0           0       500
IPCSEMNIDS                     0           0       500
IPCSHMNIDS                     0           0       500
IPCSHMPAGES                     0           0     262144 *
IPCMSGQBYTES                   ---          0     262144
IPCMSGQNUM                     ---          0     10000
IPCSHMPAGES                     ---          0       256
SHRLIBRGNSIZE                  0           0    67108864
SHRLIBMAXPAGES                  0           0       4096

```

The command displays high-water marks and current usage for an individual process when the PID=processid keyword is also specified.

```

d,omvs,l,pid=16777456
BPX0051I 14.06.28 DISPLAY OMVS 645
OMVS      000E ACTIVE          OMVS=(76)
USER      JOBNAME  ASID        PID        PPID  STATE   START   CT_SECS
STCRSE    RSED8    007E      16777456    67109106 HF----  20.00.56 113.914
LATCHWAITPID= 0 CMD=java -Ddaemon.log=/var/rdz/logs -
PROCESS LIMITS:          LIMMSG=NONE
                                CURRENT  HIGHWATER  PROCESS
                                USAGE    USAGE    LIMIT
MAXFILEPROC               83         103       256
MAXFILESIZE                ---         ---     NOLIMIT
MAXPROCUSER               97         99       200
MAXQUEUEDSIGS              0           1     1000
MAXTHREADS                9          14       200
MAXTHREADTASKS           9          14      1000
IPCSHMNSEGS                0           0       500
MAXCORESIZE                ---         ---    4194304
MAXMEMLIMIT                 0           0    16383P

```

- The **DISPLAY OMVS,PFS** operator command displays information about each physical file system that is currently part of the z/OS UNIX configuration, which includes the TCP/IP stacks.

```

d omvs,p
BPX0046I 14.35.38 DISPLAY OMVS 092
OMVS      000E ACTIVE          OMVS=(33)
PFS CONFIGURATION INFORMATION
PFS TYPE  DESCRIPTION          ENTRY      MAXSOCK  OPNSOCK  HIGHUSED
TCP      SOCKETS AF_INET      EZBPFINI   50000   244     8146
UDS        SOCKETS AF_UNIX      BPXTUINIT   64        6         10
ZFS        LOCAL FILE SYSTEM  IOEFSCM
          14:32.00 RECYCLING
HFS        LOCAL FILE SYSTEM  GFUAINIT
BPXFTCLN   CLEANUP DAEMON     BPXFTCLN
BPXFTSYN   SYNC DAEMON        BPXFTSYN
BPXFPINT   PIPE                BPXFPINT
BPXFCSIN   CHAR SPECIAL        BPXFCSIN
NFS        REMOTE FILE SYSTEM  GFSCINIT
PFS NAME   DESCRIPTION          ENTRY      STATUS   FLAGS
TCP41      SOCKETS            EZBPFINI   ACT      CD
TCP42      SOCKETS            EZBPFINI   ACT
TCP43      SOCKETS            EZBPFINI   INACT    SD
TCP44      SOCKETS            EZBPFINI   INACT

```

```
PFS PARM INFORMATION
HFS      SYNCDEFAULT(60) FIXED(50) VIRTUAL(100)
        CURRENT VALUES: FIXED(55) VIRTUAL(100)
NFS      biod(6)
```

- The **DISPLAY OMVS,PID=processid** operator command displays the thread information for a specific process.

```
d omvs,pid=16777456
BPX0040I 15.30.01 DISPLAY OMVS 637
OMVS      000E ACTIVE          OMVS=(76)
USER      JOBNAM     ASID      PID      PPID STATE   START    CT_SECS
STCRSE    RSED8      007E      16777456  67109106 HF---- 20.00.56 113.914
LATCHWAITPID=      0 CMD=java -Ddaemon.log=/var/rdz/logs -
THREAD_ID      TCB@      PRI_JOB  USERNAME  ACC_TIME SC STATE
0E08A00000000000 005E6DF0 OMVS          .927 RCV  FU
0E08F00000000001 005E6C58          .001 PTX  JYNV
0E09300000000002 005E6AC0          7.368 PTX  JYNV
0E0CB00000000008 005C2CF0 OMVS          1.872 SEL  JFNV
0E192000000003CE 005A0B70 OMVS      IBMUSER 14.088 POL  JFNV
0E18D000000003CF 005A1938      IBMUSER .581  SND  JYNV
```

Monitoring the network

When supporting a large number of clients connecting to the host, then not only Developer for System z, but also your network infrastructure must be able to handle the workload. Network management is a broad and well documented subject that falls out of the scope of Developer for System z documentation. Therefore, only the following pointers are provided.

- The **DISPLAY NET,CSM** operator command allows you to monitor the use of storage managed by the communications storage manager (CSM). You can use this command to determine how much CSM storage is in use for ECSA and data space storage pools, as documented in *Communications Server SNA Operations* (SC31-8779).

Monitoring z/OS UNIX file systems

Developer for System z uses z/OS UNIX file systems to store various types of data, such as logs and temporary files. Use the z/OS UNIX **df** command to see how many file descriptors are still available and how much free space is left before the next extent of the underlying HFS or zFS data set will be created.

```
$ df
Mounted on  Filesystem      Avail/Total  Files      Status
/tmp        (OMVS.TMP)      1393432/1396800 4294967248 Available
/u/ibmuser  (OMVS.U.IBMUSER) 1248/1728      4294967281 Available
/usr/lpp/rdz (OMVS.FEK.HHOP760) 3062/43200     4294967147 Available
/var        (OMVS.VAR)      27264/31680     4294967054 Available
```

Sample setup

The following sample setup shows the required configuration to support these requirements:

- 500 simultaneous client connections
- 300 simultaneous MVS builds (batch job)
- 200 simultaneous CARMA connections (using the CRASTART startup method)
- 3 hour inactivity time-out
- disallow usage of z/OS UNIX
- SCLM Developer Toolkit and File Manager Integration are not used
- Foresee an average Java heap usage of 5 MB

- Users have unique z/OS UNIX UIDs

Thread pool count

By default, Developer for system z tries to add 60 users to a single thread pool. However, our requirements indicate that the inactivity time-out will be active. Table 44 on page 202 shows that this will add 1 thread per connected client. This thread is a timer thread, and thus constantly active. This will prevent RSE from putting 60 users in a single thread pool, as $60 \times (16 + 1) = 1020$, and `maximum.threads` is set to 1000 by default.

We could increase `maximum.threads`, but due to the requirement to have on average 5 MB of Java heap per user, we choose to lower `maximum.clients` to 50. This keeps us within the default 256 MB maximum Java heap size ($5 \times 50 = 250$).

With 50 clients per thread pool and the need to support 500 connections, we now know we will need 10 thread pool address spaces.

Determine minimum limits

Using the formulas shown earlier in this chapter and the criteria stated at the beginning of this section, we can determine the resource usage that must be accommodated.

- Address space count - maximum
 $3 + A + N \times (x + y + z) + (2 + N \times 0.01)$
 $3 + 10 + 500 \times 1 + 200 \times 1 + 300 \times 1 + (2 + 500 \times 0.01) = 1020$
- Address space count - per user
 $x + y + z$
 $1 + 1 + 1 = 3$
- Process count - maximum
 $7 + 2 \times A + N \times (x + y + z) + (10 + N \times 0.05)$
 $7 + 2 \times 10 + 500 \times 2 + 200 \times 1 + 300 \times 0 + (10 + 500 \times 0.05) = 1562$
- Process count - per user
 $(x + y + z) + 5 \times s$
 $(2 + 1 + 0) + 5 \times 0 = 3$
- Thread count - RSE thread pool
 $9 + N \times (16 + x + y + z) + (20 + N \times 0.1)$
 $9 + 60 \times (16 + 1 + 4 + 0) + (20 + 60 \times 0.1) = 1295$
- Thread count - JES Job Monitor
 $3 + N$
 $3 + 500 = 503$
- User IDs
 $500 + 3 = 503$

The 3 extra user IDs are for STCJMON, STCLOCK and STCRSE, the Developer for System z started task user IDs.

Defining limits

Now that the resource usage numbers are known, we can customize the limiting directives with appropriate values.

- `/etc/rdz/rsed.envvars`
 - `Xmx256m`

not changed

- Dmaximum.clients=50
- Dmaximum.threads=1000

not changed

- Dminimum.threadpool.process=10
This change is optional; RSE will start new thread pools as needed
- DHIDE_ZOS_UNIX=true
- DDSTORE_IDLE_SHUTDOWN_TIMEOUT=10800000

- FEK.#CUST.PARMLIB(FEJJCNFG)
 - MAX_THREADS=503
- SYS1.PARMLIB(BPXPRMxx)
 - MAXPROCSYS(2500)

1562 minimum, added extra buffer for tasks other than Developer for System z

- MAXPROCUSER(25)

not changed, minimum 3

- MAXTHREADS(1500)

must be minimum 503 (for JES Job Monitor) if THREADSMAX in the OMVS segment of user ID STCRSE is used to set the limit for RSE (minimum 1295)

- MAXTHREADTASKS(1500)

must be minimum 503 (for JES Job Monitor) if THREADSMAX in the OMVS segment of user ID STCRSE is used to set the limit for RSE (minimum 1295)

- MAXUIDS(700)

503 minimum, added extra buffer for tasks other than Developer for System z

- MAXASSIZE(209715200)

not changed (200 MB system default), we use ASSIZEMAX in the OMVS segment of user ID STCRSE

- SYS1.PARMLIB(IEASYSxx)
 - MAXUSER=2000

1020 minimum, added extra buffer for tasks other than Developer for System z

- OMVS segment of user ID STCRSE
 - ASSIZEMAX(2147483647)

2 GB

Monitor resource usage

After activating the system limits as documented in “Defining limits” on page 221, we can start monitoring the resource usage by Developer for System z to see if adjustment of some variables is needed. Figure 58 on page 223 shows the resource

usage after 495 users logged on. (The example in the figure shows just the logging on. No user actions are indicated in the example.)

```
BPXM023I (STCRSE)
ProcessId(16779764) Memory Usage(10%) Clients(50) Order(1)
ProcessId(67108892) Memory Usage(16%) Clients(50) Order(2)
ProcessId(67108908) Memory Usage(10%) Clients(50) Order(3)
ProcessId(67108898) Memory Usage(16%) Clients(50) Order(4)
ProcessId(67108916) Memory Usage(16%) Clients(50) Order(5)
ProcessId(67108897) Memory Usage(16%) Clients(50) Order(6)
ProcessId(67108921) Memory Usage(16%) Clients(50) Order(7)
ProcessId(83886146) Memory Usage(16%) Clients(50) Order(8)
ProcessId(67108920) Memory Usage(16%) Clients(50) Order(9)
ProcessId(3622    ) Memory Usage(8%) Clients(45) Order(10)
```

Jobname	Cpu time	Storage	EXCP
-----	-----	-----	-----
JMON	1.74	43.0M	2753
LOCKD	10.05	31.9M	24621
RSED	6.65	40.1M	41780
RSED1	8.17	187.0M	76566
RSED2	13.04	184.9M	78946
RSED3	17.77	181.1M	76347
RSED4	11.63	174.9M	74638
RSED5	15.27	172.9M	72883
RSED6	13.85	180.8M	75031
RSED7	9.79	174.3M	76636
RSED8	21.59	176.1M	70583
RSED8	18.88	184.7M	76953
RSED9	9.52	189.8M	80490

Figure 58. Resource usage of sample setup

Chapter 14. 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 Extended Common Storage Area (ECSA). 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.

Developer for System z has a default of STEPLIB=NONE coded in `rsed.envvars`, as described in `rsed.envvars`, configuration file. 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.

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, such as 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 as follows:

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.

CEE.SCEELPA

The CEE.SCEELPA data set contains a subset of the LE runtime routines, which are heavily used by z/OS UNIX. You should 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 rsed.envvars, configuration file. 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 the following lines to SYS1.PARMLIB(PROGxx):

```
LPA ADD MODNAME(CEEINIT,CEEBLIBM,CEEV003,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.

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 as follows:

- When used, assign the APPC transaction to a TSO performance group.
- Assign a started task performance group (SYSSTC) to the Developer for System z server address spaces: JES Job Monitor (JMON), Lock daemon (LOCKD), RSE daemon (RSED), and RSE thread pools (RSEDx).

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

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 *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_JAVAOPTS` directive of `rsed.envvars`, as described in “Defining extra Java startup parameters with `_RSE_JAVAOPTS`” on page 37.

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

Java -Xquickstart option

Note: Java `-Xquickstart` is only useful if you use the REXEC/SSH alternate startup method for RSE server. This method is documented in “(Optional) Using REXEC (or SSH)” on page 93.

The `-Xquickstart` option can be used for improving startup time of some Java applications. `-Xquickstart` causes the JIT (Just In Time) compiler to run with a subset of optimizations; that is, a quick compile. This quick compile allows for improved startup time.

`-Xquickstart` is appropriate for shorter running applications, especially those where execution time is not concentrated into a small number of methods. `-Xquickstart` can degrade performance if it is used on longer-running applications that contain hot methods.

To enable the -Xquickstart option for the RSE server, add the following directive to the end of `rsed.envvars`:

```
_RSE_JAVA_OPTS="$ _RSE_JAVA_OPTS -Xquickstart"
```

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, add the following directive to the end of `rsed.envvars`. 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). The third statement adds the class sharing parameters to the Java startup options.

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

Note: As mentioned in “Cache security,” 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 *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 *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 the following:

- MAXSHAREPAGES, IPCSHMPAGES, 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 *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.
- A cache can be destroyed only if all JVMs using it have shut down, and the user issuing the command has sufficient permissions.

Chapter 15. CICSTS considerations

Traditionally, the role of defining resources to CICS has been the domain of the CICS administrator. There has been a reluctance to allow the application developer to define CICS resources for various reasons:

- Most CICS resource definitions have many parameters that because of their complexity, interrelationship with other resource definitions, and shop standards require CICS administrator knowledge to define correctly. Incorrect definitions can cause unexpected results that might impact the entire CICS region.
- Most customer shops provide CICS development and test environments that must be available for shared use by multiple application groups and developers. Many customer shops have Service Level Agreements in place for these environments. Meeting these agreements requires strict control of the environments.

Developer for System z addresses these issues by allowing CICS administrators to control CICS resource definition defaults, and to control the display properties of a CICS resource definition parameter by means of the CICS Resource Definition (CRD) server, which is part of Application Deployment Manager.

For example, the CICS administrator can supply certain CICS resource definition parameters that might not be updated by the application developer. Other CICS resource definition parameters may be updatable, with or without supplied defaults, or the CICS resource definition parameter can be hidden to avoid unnecessary complexity.

Once the application developer is satisfied with the CICS resource definitions they may be installed immediately in the running CICS test environment, or the definitions may be exported in a manifest for further editing and approval by a CICS administrator. The CICS administrator can use the administrative utility (batch utility) or the Manifest Processing tool to implement resource definition changes.

Note: The Manifest Processing tool is a plugin for IBM CICS Explorer.

Refer to Chapter 4, “(Optional) Application Deployment Manager,” on page 65 for more information on the tasks needed to set up Application Deployment Manager on your host system.

Customizing Application Deployment Manager adds the following services to Developer for System z:

- (on the client) IBM CICS Explorer™ provides an Eclipse-based infrastructure to view and manage CICS resources and enables greater integration between CICS tools
- (on the client) CICS Resource Definition (CRD) editor
- (on the host) CICS Resource Definition (CRD) server, which runs as a CICS application

The Application Deployment Manager CICS Resource Definition (CRD) server consists of the CRD server itself, a CRD repository, associated CICS resource definitions, and, when using the Web Service interface, Web Service bind files, and

a sample pipeline message handler. The CRD server must run in a Web Owning Region (WOR), which is referenced in the Developer for System z documentation as the CICS primary connection region.

Refer to the Developer for System z Information Center (<http://publib.boulder.ibm.com/infocenter/ratdevz/v7r6/index.jsp>) to learn more about the services Application Deployment Manager available in the current release of Developer for System z.

RESTful versus Web Service

CICS Transaction Server provides in version 4.1 and higher support for an HTTP interface designed using Representational State Transfer (RESTful) principles. This RESTful interface is now the strategic CICSTS interface for use by client applications. The older Web Service interface has been stabilized, and enhancements will be for the RESTful interface only.

Application Deployment Manager follows this statement of direction and requires the RESTful CRD server for all services that are new to Developer for System version 7.6 or higher.

The RESTful and Web Service interfaces can be active concurrently in a single CICS region, if desired. In this case, there will be two CRD servers active in the region. Both servers will share the same CRD repository. Note that CICS will issue some warnings about duplicate definitions when the second interface is defined to the region.

Primary versus non-primary connection regions

A CICS test environment may consist of several Multi-Region Option (MRO) connected regions. Over time, unofficial designations have been used to categorize these regions. Typical designations are Terminal Owning Region (TOR), Web Owning Region (WOR), Application Owning Region (AOR), and Data Owning Region (DOR).

A Web Owning Region is used to implement CICS Web Services support, and the Application Deployment Manager CICS Resource Definition (CRD) server must run in this region. This region is known to Application Deployment Manager as the CICS primary connection region. The CRD client implements a Web service connection to the CICS primary connection region.

CICS non-primary connection regions are all other regions that the CRD server can service. This service includes viewing resources using IBM CICS Explorer and defining resources using the CICS resource definition editor.

If CICSplex SM Business Application Services (BAS) is used to manage the CICS resource definitions of the CICS primary connection region, then all other CICS regions managed by BAS can be serviced by the CRD server.

CICS regions not managed by BAS require additional changes to be serviceable by the CRD server.

CICS resource install logging

Actions done by the CRD server against the CICS resources are logged in the CICS CSDL TD queue, which typically points to DD MSGUSR of your CICS region.

If CICSplex SM Business Application Services (BAS) is used to manage your CICS resource definitions, then the CICSplex SM EYUPARM directive BASLOGMSG must be set to (YES) for the logging to be created.

Application Deployment Manager security

CRD repository security

The CRD server repository VSAM data set holds all the default resource definitions and must therefore be protected against updates, but developers must be allowed to read the values stored here. Refer to “Define data set profiles” on page 166 for sample RACF commands to protect the CRD repository.

Pipeline security

When a SOAP message is received by CICS through the Web Service interface, the message is processed by a pipeline. A pipeline is a set of message handlers that are executed in sequence. CICS reads the pipeline configuration file to determine which message handlers should be invoked in the pipeline. A message handler is a program in which you can perform special processing of Web service requests and responses.

Application Deployment Manager provides a sample pipeline configuration file that specifies the invocation of a message handler and a SOAP header processing program.

The pipeline message handler (ADNTMSGH) is used for security by processing the user ID and password in the SOAP header. ADNTMSGH is referenced by the sample pipeline configuration file and must therefore be placed into the CICS RPL concatenation.

Transaction security

CPIH is the default transaction ID under which an application invoked by a pipeline will run. Typically, CPIH is set for a minimal level of authorization.

Developer for System z supplies multiple transactions that are used by the CRD server when defining and inquiring CICS resources. These transaction IDs are set by the CRD server, depending on the requested operation. Refer to Chapter 4, “(Optional) Application Deployment Manager,” on page 65 for more information on customizing the transaction IDs.

Transaction	Description
ADMS	For requests from the Manifest Processing tool to change CICS resources. Typically, this is intended for CICS administrators. This transaction requires a high level of authorization.
ADMI	For requests that define, install or uninstall CICS resources. This transaction might require a medium level of authorization, depending on your site policies.
ADMR	For all other requests that retrieve CICS environmental or resource information. This transaction might require a minimal level of authorization, depending on your site policies.

Some, or all, of the resource definition requests done by the CRD server transactions should be secured. At a minimum, the update commands (update default Web service parameters, default descriptor parameters, and file name to data set name binding) should be secured to prevent all but CICS administrators from issuing these commands used to set global resource defaults.

When the transaction is attached, CICS resource security checking, if enabled, insures that the user ID is authorized to run the transaction ID.

Resource checking is controlled by the RESSEC option in the transaction that is running, the RESSEC system initialization parameter, and for the CRD server, the XPCT system initialization parameter.

Resource checking occurs only if the XPCT system initialization parameter has a value other than NO and either the RESSEC option in the TRANSACTION definition is YES or the RESSEC system initialization parameter is ALWAYS.

The following RACF commands give a sample on how the CRD server transactions can be protected. Refer to *RACF Security Guide for CICSTS* for more information on defining CICS security.

- RALTER GCICSTRN SYSADM UACC(NONE) ADDMEM(ADMS)
- PERMIT SYSADM CLASS(GCICSTRN) ID(#cicsadmin)
- RALTER GCICSTRN DEVELOPER UACC(NONE) ADDMEM(ADMI)
- PERMIT DEVELOPER CLASS(GCICSTRN) ID(#cicsdeveloper)
- RALTER GCICSTRN ALLUSER UACC(READ) ADDMEM(ADMR)
- SETROPTS RACLIST(TCICSTRN) REFRESH

SSL encrypted communication

SSL encryption of the data stream is supported when the Application Deployment Manager client uses the Web Services interface to invoke the CRD server. The usage of SSL for this communication is controlled by the SSL(YES) keyword in the CICSTS TCIPSERVICE definition, as documented in *RACF Security Guide for CICSTS*.

Resource security

CICSTS provides the ability to protect resources and the commands to manipulate them. Certain Application Deployment Manager actions might fail if security is active, but not configured completely (for example, granting permissions to manipulate new resource types).

Upon function failure in Application Deployment Manager, examine the CICS log for messages like the following, and take corrective action, as documented in *RACF Security Guide for CICSTS*.

```
DFHXS1111 %date %time %applid %tranid Security violation by user
%userid at netname %portname for resource %resource in class
%classname. SAF codes are (X'safresp',X'safreas'). ESM codes are
(X'esmresp',X'esmreas').
```

Administrative utility

Developer for System z provides the administrative utility to let CICS administrators provide the default values for CICS resource definitions. These defaults can be read-only, or can be editable by the application developer.

The administrative utility provides the following functions:

- CICSplex name for CICSplex managed test environments
- CICSplex SM staging group name
- Manifest export rule setting
- CICS resource attribute defaults and display permissions
- CICS logical to physical binding used for VSAM data set definitions

The administrative utility is invoked by sample job ADNJSPAU in data set FEK.#CUST.JCL. The usage of this utility requires UPDATE access to the CRD repository.

ADNJSPAU is located in FEK.#CUST.JCL, unless the z/OS system programmer specified a different location when he customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details.

Note: The CRD repository must be closed in CICS before running the ADNJSPAU job. The repository can be opened again after job completion. For example, after signing on to CICS, enter the following commands to close and open the file, respectively:

- CEMT S FILE(ADNREPF0) CLOSED
- CEMT S FILE(ADNREPF0) OPEN

Input control statements are used to update the CRD repository for a CICS test environment, for which the following general syntax rules apply:

- An asterisk in position 1 indicates a comment line.
- A DEFINE command must begin in position 1, followed by a single space, followed by a valid keyword, such as TRANSACTION.
- A keyword value must immediately follow a keyword. No intervening spaces are permitted. The only exception is for display permission keywords UPDATE, PROTECT, and HIDDEN, which have no values.
- Keyword values are enclosed within parenthesis.
- A keyword and its value must be contained on a single line.

The following sample definitions follow the structure of the DFHCSDUP commands, as defined in the *CICS Resource Definition Guide for CICSTS*. The only difference is the insertion of the following display permission keywords used to group the attribute values into three permission sets:

UPDATE	Attributes following this keyword will be updatable by an application developer using Developer for System z. This is also the default for omitted attributes.
PROTECT	Attributes following this keyword will display, but be protected from update by an application developer using Developer for System z.

HIDDEN	Attributes following this keyword will not display, and will be protected from update by an application developer using Developer for System z.
--------	---

See the following ADNJSAPU code sample.

```

//ADNJSPAU JOB <JOB PARAMETERS>
//*
//ADNSPAU EXEC PGM=ADNSPAU,REGION=1M
//STEPLIB DD DISP=SHR,DSN=FEK.SFEKLOAD
//ADMREP DD DISP=OLD,DSN=FEK.#CUST.ADNREP0
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
*
* CICSplex SM parameters
*
DEFINE CPSMNAME( )
*DEFINE STAGINGGROUPNAME(ADMSTAGE)
*
* Manifest export rule
*
DEFINE MANIFESTEXPORTRULE(installOnly)
*
* CICS resource definition defaults
* Omitted attributes default to UPDATE.
*
* DB2TRAN default attributes
*
DEFINE DB2TRAN()
    UPDATE DESCRIPTION()
    ENTRY()
    TRANSID()
*
* DOCTEMPLATE default attributes
*
DEFINE DOCTEMPLATE()
    UPDATE DESCRIPTION()
    TEMPLATENAME()
    FILE() TSQUEUE() TDQUEUE() PROGRAM() EXITPGM()
    DDNAME(DFHHTML) MEMBERNAME()
    HFSFILE()
    APPENDCRLF(YES) TYPE(EBCDIC)
*
* File default attributes
*
DEFINE FILE()
    UPDATE DESCRIPTION()
    RECORDSIZE() KEYLENGTH()
    RECORDFORMAT(V) ADD(NO)
    BROWSE(NO) DELETE(NO) READ(YES) UPDATE(NO)
    REMOTESYSTEM() REMOTENAME()
    PROTECT DSNNAME() RLSACCESS(NO) LSRPOOLID(1) STRINGS(1)
    STATUS(ENABLED) OPENTIME(FIRSTREF)
    DISPOSITION(SHARE) DATABUFFERS(2) INDEXBUFFERS(1)
    TABLE(NO) MAXNUMRECS(NOLIMIT)
    READINTEG(UNCOMMITTED) DSNSHARING(ALLREQS)
    UPDATEMODEL(LOCKING) LOAD(NO)
    JNLREAD(NONE) JOURNAL(NO)
    JNLSYNCREAD(NO) JNLUPDATE(NO)
    JNLADD(NONE) JNLSYNCSWRITE(YES)
    RECOVERY(NONE) FWDRECOVLOG(NO)
    BACKUPTYPE(STATIC)
    PASSWORD() NSRGROUP()
    CFDTPOOL() TABLENAME()

```

Figure 59. ADNJSPAU - CICS administrative utility (Part 1 of 3)

```

*
* Mapset default attributes
*
DEFINE MAPSET()
    UPDATE  DESCRIPTION()
    PROTECT RESIDENT(NO) STATUS(ENABLED)
           USAGE(NORMAL) USELPACOPY(NO)
** Processtype default attributes
*
DEFINE PROCESSTYPE()
    UPDATE  DESCRIPTION()
           FILE(BTS)
    PROTECT STATUS(ENABLED)
           AUDITLOG() AUDITLEVEL(OFF)
*
* Program default attributes
*
DEFINE PROGRAM()
    UPDATE  DESCRIPTION()
           CEDF(YES) LANGUAGE(LE370)
           REMOTESYSTEM() REMOTENAME() TRANSID()
    PROTECT API(CICSAPI) CONCURRENCY(QUASIRENT)
           DATALOCATION(ANY) DYNAMIC(NO)
           EXECCKEY(USER) EXECUTIONSET(FULLAPI)
           RELOAD(NO) RESIDENT(NO)
           STATUS(ENABLED) USAGE(NORMAL) USELPACOPY(NO)
    HIDDEN JVM(NO) JVMCLASS() JVMPROFILE(DFHJVMPR)
*
* TDQueue default attributes
*
DEFINE TDQUEUE()
    UPDATE  DESCRIPTION()
           TYPE(INTRA)
* Extra partition parameters
    DDNAME() DSNAME()
    REMOTENAME() REMOTESYSTEM() REMOTELength(1)
    RECORDSIZE() BLOCKSIZE(0) RECORDFORMAT(UNDEFINED)
    BLOCKFORMAT() PRINTCONTROL() DISPOSITION(SHR)
* Intra partition parameters
    FACILITYID() TRANSID() TRIGERRLEVEL(1)
    USERID()
* Indirect parameters
    INDIRECTNAME()
    PROTECT WAIT(YES) WAITACTION(REJECT)
* Extra partition parameters
    DATABUFFERS(1)
    SYSOUTCLASS() ERROROPTION(IGNORE)
    OPENTIME(INITIAL) REWIND(LEAVE) TYPEFILE(INPUT)
* Intra partition parameters
    ATIFACILITY(TERMINAL) RECOVSTATUS(NO)

```

Figure 59. ADNJSPAU - CICSTS administrative utility (Part 2 of 3)

```

*
* Transaction default attributes
*
DEFINE TRANSACTION()
    UPDATE  DESCRIPTION()
            PROGRAM()
            TWASIZE(0)
            REMOTESYSTEM() REMOTENAME() LOCALQ(NO)
    PROTECT PARTITIONSET() PROFILE(DFHCICST)
            DYNAMIC(NO) ROUTABLE(NO)
            ISOLATE(YES) STATUS(ENABLED)
            RUNAWAY(SYSTEM) STORAGECLEAR(NO)
            SHUTDOWN(DISABLED)
            TASKDATAKEY(USER) TASKDATALOC(ANY)
            BREXIT() PRIORITY(1) TRANCLASS(DFHTCL00)
            DTIMOUT(NO) RESTART(NO) SPURGE(NO) TPURGE(NO)
            DUMP(YES) TRACE(YES) CONFDATA(NO)
            OTSTIMEOUT(NO) WAIT(YES) WAITTIME(00,00,00)
            ACTION(BACKOUT) INDOUBT(BACKOUT)
            RESSEC(NO) CMDSEC(NO)
            TRPROF()
            ALIAS() TASKREQ()
            XTRANID() TPNAME() XTPNAME()
|
| *
| * URDIMAP attributes
| *
| DEFINE URIMAP()
|     UPDATE  USAGE(CLIENT)
|             DESCRIPTION()
|             PATH(/required/path)
|             TCPIPSERVICE()
|             TRANSACTION()
|             PROGRAM()
|     PROTECT ANALYZER(NOANALYZER)
|             ATOMSERVICE()
|             CERTIFICATE()
|             CHARACTERSET()
|             CIPHERS()
|             CONVERTER()
|             HFSFILE()
|             HOST(host.mycompany.com)
|             HOSTCODEPAGE()
|             LOCATION()
|             MEDIATYPE()
|             PIPELINE()
|             PORT(NO)
|             REDIRECTTYPE(NONE)
|             SCHEME(HTTP)
|             STATUS(ENABLED)
|             TEMPLATENAME()
|             USERID()
|             WEBSERVICE()
|
| *
| * Optional file name to VSAM data set name binding
| *
| *DEFINE DSBINDING() DSNAME()
| /*

```

Figure 59. ADNJSPAU - CICSTS administrative utility (Part 3 of 3)

Administrative utility migration notes

Developer for System z version 7.6.1 added URIMAP support to the Administrative utility. To be able to use the URIMAP support, the CRD repository

VSAM data set must be allocated with a maximum record size of 3000. Up till Developer for System z version 7.6.1, the sample CRD repository allocation job uses a maximum record size of 2000.

Follow these steps to enable the URIMAP support if you're using an older CRD repository:

1. Create a backup of your existing CRD repository, FEK.#CUST.ADNREPF0.
2. Delete the existing CRD repository.
3. Customize and submit job FEK.SFEKSAMP(ADNVCRD) to allocate and initialize a new CRD repository. Refer to the documentation within the member for customization instructions.
4. Customize and submit job FEK.SFEKSAMP(ADNJSPAU) to use the Administrative utility to populate the new CRD repository.

Note:

- Migrating the existing CRD repository is not necessary, because the Administrative utility replaces the complete contents of the CRD repository each time it is executed.
- There are no version compatibility issues with the CRD repository. All supported Developer for System z client and host code will work with either maximum record size. But URIMAP support will be disabled if the maximum record size is not 3000.

Administrative utility messages

The following messages are issued by the Administrative utility to the SYSPRINT DD. Messages CRAZ1803E, CRAZ1891E, CRAZ1892E, and CRAZ1893E contain file status, VSAM return, VSAM function, and VSAM feedback codes. VSAM return, function, and feedback codes are documented in *DFSMS Macro Instructions for Data Sets* (SC26-7408). File status codes are documented in *Enterprise COBOL for z/OS Language Reference* (SC27-1408).

CRAZ1800I

completed successfully on line <last control statement line number>

Explanation: The system programmer administrative utility completed successfully.

User response: None.

CRAZ1801W

completed with warnings on line <last control statement line number>

Explanation: The system programmer administrative utility completed with one or more warnings found when processing control statements.

User response: Check other warning messages.

CRAZ1802E

encountered an error on line < line number>

Explanation: The system programmer administrative utility encountered a severe error.

User response: Check other warning messages.

CRAZ1803E

Repository open error, status=<file status code> RC=<VSAM return code> FC=<VSAM function code> FB=<VSAM feedback code>

Explanation: The system programmer administrative utility encountered a severe error opening the CRD repository.

User response: Check VSAM status, return, function, and feedback codes.

CRAZ1804E

Unrecognized input record on line <line number>

Explanation: The system programmer administrative utility encountered an unrecognized input control statement.

User response: Check a **DEFINE** command was followed by a single space, followed by the keyword CPSMNAME, STAGINGGROUPNAME, MANIFESTEXPORTRULE, DSBINDING, DB2TRAN, DOCTEMPLATE, FILE, MAPSET, PROCESSTYPE, PROGRAM, TDQUEUE, or TRANSACTION.

CRAZ1805E

Processing keyword <keyword> on line <line number>

Explanation: The system programmer administrative utility is processing the **DEFINE** keyword input control statement.

User response: None.

CRAZ1806E

Invalid manifest export rule on line <line number>

Explanation: The system programmer administrative utility encountered an invalid manifest export rule.

User response: Check that the **MANIFESTEXPORTRULE** keyword value is "installOnly", "exportOnly", or "both".

CRAZ1807E

Missing DSNNAME keyword on line <line number>

Explanation: The system programmer administrative utility was processing a **DEFINE DSBINDING** control statement which is missing the **DSNAME** keyword.

User response: Check that the **DEFINE DSBINDING** control statement contains the **DSNAME** keyword.

CRAZ1808E

Invalid keyword value for keyword <keyword> on line <line number>

Explanation: The system programmer administrative utility was processing a **DEFINE** control statement and encountered an invalid value for the named keyword.

User response: Check that the length and value of the named keyword is correct.

CRAZ1890W

Keyword syntax error on line <line number>

Explanation: The system programmer administrative utility was processing a **DEFINE** control statement and encountered a syntax error for a keyword or keyword value.

User response: Check that the keyword value is enclosed in parenthesis and immediately follows the keyword. The keyword and keyword value must both be contained on the same line.

CRAZ1891W

**Repository duplicate key write error, status=<file status code>
RC=<VSAM return code> FC=<VSAM function code> FB=<VSAM
feedback code>**

Explanation: The system programmer administrative utility encountered a duplicate key error writing to the CRD repository.

User response: Check VSAM status, return, function, and feedback codes.

CRAZ1892W

**Repository write error, status=<file status code> RC=<VSAM return
code> FC=<VSAM function code> FB=<VSAM feedback code>**

Explanation: The system programmer administrative utility encountered a severe error writing to the CRD repository.

User response: Check VSAM status, return, function, and feedback codes.

CRAZ1893W

**Repository read error, status=<file status code> RC=<VSAM return
code> FC=<VSAM function code> FB=<VSAM feedback code>**

Explanation: The system programmer administrative utility encountered a severe error reading from the CRD repository.

User response: Check VSAM status, return, function, and feedback codes.

Chapter 16. Customizing the TSO environment

This appendix is provided to assist you with mimicking a TSO logon procedure by adding DD statements and data sets to the TSO environment in Developer for System z.

The TSO Commands service

The TSO Commands service is the Developer for System z component which executes TSO and (batch) ISPF commands, and returns the result to the requesting client. These commands can be requested implicitly by the product, or explicitly by the user.

The sample members provided with Developer for System z create a minimal TSO/ISPF environment. If the developers in your shop need access to custom or third-party libraries, the z/OS system programmer must add the necessary DD statements and libraries to the TSO Commands service environment. Although the implementation is different in Developer for System z, the logic behind it is identical to the TSO logon procedure.

Note: The TSO Commands service is a non-interactive command-line tool, so commands or procedures that prompt for data or display ISPF panels will not work. A 3270 emulator, such as the Host Connect Emulator which is part of the Developer for System z client, is needed to execute these.

Access methods

Since version 7.1, Developer for System z provides a choice on how to access the TSO Commands service.

- ISPF's TSO/ISPF Client Gateway service, which requires a minimum ISPF service level. This is the default method used in the provided samples.
- An APPC transaction (as in pre-version 7.1 releases).

Note: ISPF's TSO/ISPF Client Gateway service replaces the SCLM Developer Toolkit function used in version 7.1.

Check `rsed.envvars` to determine which access method is used for version 7.1 and higher hosts. If defaults were used during the configuration process, `rsed.envvars` resides in `/etc/rdz/`.

- If the `_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DTSO_SERVER=APPC"` statement is not present (or commented out, which is the default), ISPF's TSO/ISPF Client Gateway service is used.
- If the `_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DTSO_SERVER=APPC"` statement is present (and not commented out), APPC is used.

Using the TSO/ISPF Client Gateway access method

Basic customization – ISPF.conf

The `ISPF.conf` configuration file (by default located in `/etc/rdz/`) defines the TSO/ISPF environment used by Developer for System z. There is only one active `ISPF.conf` configuration file, which is used by all Developer for System z users.

The main section of the configuration file defines the DD names and the related data set concatenations, like that in the following sample:

```
sysproc=ISP.SISPCLIB,FEK.SFEKPROC
ispmlib=ISP.SISPMENU
isptlib=ISP.SISPTENU
ispplib=ISP.SISPPENU
ispplib=ISP.SISPSLIB
ispllib=ISP.SISPLOAD
myDD=HLQ1.LLQ1,HLQ2.LLQ2
```

- Each DD definition uses exactly one line (multi-line is not supported), and there are no line length limits.
- The definitions are not case sensitive, and any white space will be ignored.
- Comment lines start with an asterisk (*).
- DD names are followed by an equal sign (=), which in turn is followed by the data set concatenation. Multiple data set names are separated by a comma (,).
- Data set concatenations are searched in the order they are listed.
- Data sets must be fully qualified, without being enclosed in quotes ('), and without the use of variables.
- All data sets are allocated with DISP=SHR.
- New DD names can be added at will, but must obey the (JCL) rules for DD names and may not conflict with other configuration parameters in ISPF.conf. Also, ISPPROF is allocated dynamically (DISP=NEW,DELETE) by the TSO/ISPF Client Gateway service.

Advanced – Use existing ISPF profiles

By default, the TSO/ISPF Client Gateway creates a temporary ISPF profile for the TSO Commands service. However, you can instruct the TSO/ISPF Client Gateway to use a copy of an existing ISPF profile. The key here is the `_RSE_CMDSERV_OPTS` statement in `rsed.envvars`.

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

Uncomment the statement (remove the leading pound sign (#) character) and provide the fully qualified data set name of the existing ISPF profile to use this facility.

The following variables can be used in the data set name:

- `&SYSUID.` to substitute the developer's user ID
- `&SYSPREF.` to substitute the developer's TSO prefix

Note:

- If the data set name passed in "ISPPROF" is invalid, a temporary, empty ISPF profile is used instead.
- The ISPF profile (both temporary and copied) is deleted at the end of the session. Changes made to the profile are not merged into the existing ISPF profile.

Advanced – Using an allocation exec

The `allocjob` statement in `ISPF.conf` (which is commented out by default) points to an exec which can be used to provide further data set allocations by user ID.

```
*allocjob = FEK.#CUST.CNTL(CRAISPRX)
```

Uncomment the statement (remove the leading asterisk (*) character) and provide the fully qualified reference to the allocation exec to use this facility.

- The exec is executed after allocation of ISPPROF and the DDs defined in ISPF.conf, but before ISPF is initialized. Ensure that your allocation exec does not undo these definitions.
- 1 parameter is passed to the exec; the user ID of the caller.
- A sample exec CRAISPRX is provided in sample library FEK.#CUST.CNTL, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See “Customization setup” on page 13 for more details.

Note: As the exec is called before ISPF is initialized, you cannot use **VPUT** and **VGET**. You can however create your own implementation of these functions using a PDS(E) or VSAM file.

Advanced – Use multiple allocation execs

Although ISPF.conf only supports calling one allocation exec, there are no limits on that exec calling another exec. And the user ID of the client being passed as parameter opens the door to calling personalized allocation execs. You can, for example, check if member USERID'.EXEC(ALLOC)' exists and execute it.

An elaborate variation to this theme enables you to use the existing TSO logon procedures, as follows:

- Read a user-specific configuration file, such as USERID'.FEKPROF'.
- See which logon procedure is mentioned in the file.
- Read the mentioned procedure from SYS1.PROCLIB and parse it to find the DD statements and data set allocations within.
- Allocate the data set in a similar fashion as the real logon procedure.

Advanced – Multiple ISPF.conf files with multiple Developer for System z setups

If the allocation exec scenarios described above cannot handle your specific needs, you can create different instances of Developer for System z's RSE communication server, each using their own ISPF.conf file. The main drawback of the method described below is that Developer for System z users must connect to different servers on the same host to get the desired TSO environment.

Note: Creating a second instance of the RSE server only requires duplicating and updating configuration files, startup JCL and started task definitions. A new installation of the product is not necessary, nor is any code duplicated.

```
$ cd /etc/rdz
$ mkdir /etc/rdz/tso2
$ cp rsed.envvars /etc/rdz/tso2
$ cp ISPF.conf /etc/rdz/tso2
$ ls /etc/rdz/tso2
ISPF.conf          rsed.envvars
$ oedit /etc/rdz/tso2/rsed.envvars
-> change: _CMDSEV_CONF_HOME=/etc/rdz/tso2
-> uncomment and change: -Ddaemon.log=/var/rdz/logs/tso2
-> add at the END:
# -- NEEDED TO FIND THE REMAINING CONFIGURATION FILES
CFG_BASE=/etc/rdz
CLASSPATH=.:$CFG_BASE:$CLASSPATH
# --
$ oedit /etc/rdz/tso2/ISPF.conf
-> change: change as needed
```

The commands in the previous example copy the Developer for System z configuration files that require changes to a newly created tso2 directory. The _CMDSESV_CONF_HOME variable in rsed.envvars must be updated to define the new ISPF.conf home directory and daemon.log must be updated to define a new log location (which is created automatically if it does not exist). The CLASSPATH update ensures that RSE can find the configuration files that were not copied to tso2. The ISPF.conf file itself can be updated to match your needs. Note that the ISPF workarea (variable _CMDSESV_WORK_HOME in rsed.envvars) can be shared among both instances.

What is left now is creating a new started task for RSE that uses a new port number and the new /etc/rdz/tso2 configuration files.

Refer to the related sections in this publication for more information on the actions shown above.

Using the APPC access method

Basic customization – APPC transaction JCL

The definition of an APPC transaction consists of APPC parameters and a transaction JCL. The sample JCL to create a Developer for System z APPC transaction, FEK.#CUST.JCL(FEKAPPC), holds two options to define the transaction JCL, with and without ISPF support.

```
//SYSIN DD DDNAME=SYSINISP * use SYSINTSO or SYSINISP
```

The client gets the TSO/ISPF environment defined in the transaction JCL, so by customizing this section, following regular DD rules, you can customize the environment for the client.

```
...
//CMDSESV EXEC PGM=IKJEFT01,DYNAMNBR=50,
// PARM='ISPSTART CMD(%FEKFRSRV TIMEOUT=60) NEWAPPL(ISR) NESTMACS'
//SYSPROC DD DISP=SHR,DSN=FEK.SFEKPROC
//ISPPLIB DD DISP=SHR,DSN=ISP.SISPPENU
//ISPMLIB DD DISP=SHR,DSN=ISP.SISPMENU
//ISPTLIB DD DISP=SHR,DSN=ISP.SISPTENU
//ISPSLIB DD DISP=SHR,DSN=ISP.SISPSENU
//ISPPROF DD DISP=(NEW,DELETE,DELETE),UNIT=SYSALLDA,
// SPACE=(TRK,(1,1,5)),LRECL=80,RECFM=FB
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSIN DD DUMMY
//MYDD DD DISP=SHR,DSN=HLQ1.LLQ1
// DISP=SHR,DSN=HLQ2.LLQ2
```

Note: An existing APPC transaction can be modified using the APPC ISPF panels.

Advanced – Use existing ISPF profiles

If ISPF support is selected, Developer for System z creates by default a temporary ISPF profile for the TSO Commands service. However, you can instruct Developer for System z to use a copy of an existing ISPF profile. As described in the FEK.SFEKSAMP(FEKAPPC) sample job, you must perform the following:

- Uncomment the COPY step in the transaction JCL (EXEC and related DD cards).
- Change &SYSUID..ISPPROF to match the user's ISPF profile data set name.
- Comment out the first ISPPROF DD statement in the CMDSESV step, and uncomment the second one.

```

...
//COPY      EXEC PGM=IEBCOPY      * (optional) clone existing ISPF profile
//SYSPRINT DD SYSOUT=*
//SYSUT1    DD DISP=SHR,DSN=&SYSUID..ISPROF
//SYSUT2    DD DISP=(MOD,PASS),DSN=&&PROF,
//           UNIT=SYSALLDA,
//           LIKE=&SYSUID..ISPROF
//*
//CMSERV    EXEC PGM=IKJEFT01,DYNAMNBR=50,
//           PARM='ISPSTART CMD(%FEKFRSRV TIMEOUT=60) NEWAPPL(ISR) NESTMACS'
//*ISPPROF DD DISP=(NEW,DELETE,DELETE),UNIT=SYSALLDA,
//           SPACE=(TRK,(1,1,5)),LRECL=80,RECFM=FB
//ISPPROF DD DISP=(OLD,DELETE,DELETE),DSN=&&PROF

```

Note: If an invalid data set name is used, the startup of the APPC transaction (and thus the TSO Commands service) will fail.

Advanced – Using an allocation exec

The sample transaction JCL calls the TSO Commands service directly by passing its name (FEKFRSRV) as parameter to program IKJEFT01. You can change this to call another exec. This exec can do allocations based on the current user ID and then call the TSO Commands service.

Contrary to the TSO/ISPF Client Gateway access method, variables stored in the user's ISPF profile can be used by this exec to assist in customizing the environment. But keep in mind that updates to the profile will be lost at session end since you are using a temporary copy, not the actual profile.

Note, however, that the use of an allocation exec in the APPC transaction is not supported and the description above is as-is.

Advanced – Multiple APPC transactions with multiple Developer for System z setups

If you need multiple unique TSO environments, you can create different instances of Developer for System z's RSE communication server, each using their own APPC transaction. The main drawback of the method described below is that Developer for System z users must connect to different servers on the same host to get the desired TSO environment.

Note: Creating a second instance of the RSE server only requires duplicating and updating configuration files, startup JCL and started task definitions. A new installation of the product is not necessary, nor is any code duplicated.

```

$ cd /etc/rdz
$ mkdir /etc/rdz/tso2
$ cp rsed.envvars /etc/rdz/tso2
$ ls /etc/rdz/tso2/
rsed.envvars
$ oedit /etc/rdz/tso2/rsed.envvars
-> uncomment and change: _FEKFSCMD_TP_NAME_=FEKFTS02
-> uncomment and change: -Ddaemon.log=/var/rdz/logs/tso2
-> add at the END:
# -- NEEDED TO FIND THE REMAINING CONFIGURATION FILES
CFG_BASE=/etc/rdz
CLASSPATH=.:$CFG_BASE:$CLASSPATH
# --

```

The commands above create a new tso2 directory and copy the Developer for System z configuration files that require changes to the new location. The `_FEKFSCMD_TP_NAME_` variable in `rsed.envvars` must be updated to define the new

APPC transaction name, and daemon.log must be updated to define a new daemon log location (which is created automatically if it does not exist). The CLASSPATH update ensures that RSE can find the configuration files that were not copied to tso2.

```
//FEKAPPCC JOB CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1),NOTIFY=&SYSUID
/*
//TPADD EXEC PGM=ATBSDFMU
//SYSSDLIB DD DISP=SHR,DSN=SYS1.APPCTP
//SYSSDOUT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN DD DDNAME=SYSINISP * use SYSINTSO or SYSINISP
//SYSINISP DD DATA,DLM='QT'
TPADD
TPNAME(FEKFTS02)
ACTIVE(YES)
TPSCHED_DELIMITER(DLM1)
KEEP_MESSAGE_LOG(ERROR)
MESSAGE_DATA_SET(&SYSUID..FEKFTS02.&TPDATE..&TPTIME..LOG)
DATASET_STATUS(MOD)
CLASS(A)
JCL_DELIMITER(DLM2)
//FEKFTS02 JOB CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1)
/*
//CMDSEV EXEC PGM=IKJEFT01,DYNAMNBR=50,
// PARM='ISPSTART CMD(%FEKFRSRV TIMEOUT=60) NEWAPPL(ISR) NESTMACS'
//SYSPROC DD DISP=SHR,DSN=FEK.SFEKPROC
//ISPPLIB DD DISP=SHR,DSN=ISP.SISPPENU
//ISPMLIB DD DISP=SHR,DSN=ISP.SISPMENU
//ISPTLIB DD DISP=SHR,DSN=ISP.SISPTENU
//ISPSLIB DD DISP=SHR,DSN=ISP.SISPSENU
//ISPPROF DD DISP=(NEW,DELETE,DELETE),UNIT=SYSALLDA,
// SPACE=(TRK,(1,1,5)),LRECL=80,RECFM=FB
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSIN DD DUMMY
//MYDD DD DISP=SHR,DSN=HLQ1.LLQ1
// DISP=SHR,DSN=HLQ2.LLQ2
DLM2
DLM1
QT
```

Figure 60. FEKAPPCC - create a second APPC transaction

Next, create a new APPC transaction by customizing and submitting sample job FEK.#CUST.JCL(FEKAPPCC), as shown in the sample above. On top of the normal customization (described in the JCL) you must also change the TPNAME to TPNAME(FEKFTS02) to match the _FEKFSCMD_TP_NAME_ definition in the new rsed.envvars. You should also change the name in the MESSAGE_DATA_SET variable and the JOB name of the transaction JCL.

What is left now is creating a new started task for RSE that uses a new port number and the new /etc/rdz/tso2 configuration files.

Refer to the related sections in this publication for more information on the actions shown above.

Chapter 17. Running multiple instances

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 such as 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 two (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.

Note:

- FEK and /usr/lpp/rdz are the high level qualifier and path used during the installation of the product. FEK.#CUST, /etc/rdz and /var/rdz are the default locations used during the customization of the product (see “Customization setup” on page 13 for more information).
- You should install Developer for System z in a private file system (HFS or zFS) to ease deploying the z/OS UNIX parts of the product.
- If you can not use a private file system, you should use an archiving tool such as the z/OS UNIX tar command to transport the z/OS UNIX directories from system to system. This to preserve the attributes (such as program control) for the Developer for System z files and directories.

Refer to *UNIX System Services Command Reference* (SA22-7802) for more information on the following sample commands to archive and restore the Developer for System z installation directory.

- Archive: `cd /SYS1/usr/lpp/rdz; tar -cSf /u/userid/rdz.tar`
- Restore: `cd /SYS2/usr/lpp/rdz; tar -xSf /u/userid/rdz.tar`

Identical setup across a sysplex

Developer for System z configuration files (and code) can be shared across different systems in a sysplex, with each system running its own identical copy of Developer for System z, if a few guidelines are obeyed.

- The log files should end up in unique locations to avoid one system overwriting information from another. By routing the z/OS UNIX logs to specific locations with the `daemon.log` and `user.log` directives in `rsed.envvars`, you can share the configuration files if you mount a system specific z/OS UNIX file system on the specified path. This way, all logs are written to the same logical place, but due to the unshared file system underneath, they end up in different physical locations.
- Configuration-type directories like `/etc/rdz/` and `/var/rdz/projects/` can be shared across the sysplex, as Developer for System z uses them in read-only mode.
- Temporary data directories like `/tmp/` and `/var/rdz/WORKAREA/` should be unique per system, as temporary file names are not sysplex-aware.

- If you share the code, you should also share the configuration files to ensure you do not have some systems that are out of synchronization after applying maintenance.

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 CANNOT 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, and ports to avoid overlapping the current setup.

In the SSL sample mentioned above, the current RSE daemon setup can be cloned, after which the cloned setup can be updated. Next the RSE daemon startup JCL can be cloned and customized with a new TCP/IP port and the location of the updated configuration files. The MVS customizations (JES Job Monitor, and so on) can be shared between the SSL and non-SSL instances. This would result in the following actions:

```
$ cd /etc/rdz
$ mkdir /etc/rdz/ssl
$ cp rsed.envvars /etc/rdz/ssl
$ cp ssl.properties /etc/rdz/ssl
$ ls /etc/rdz/ssl/
rsed.envvars    ssl.properties
$ oedit /etc/rdz/ssl/rsed.envvars
  -> uncomment and change: -Ddaemon.log=/var/rdz/logs/ssl
  -> add at the END:
      # -- NEEDED TO FIND THE REMAINING CONFIGURATION FILES
      CFG_BASE=/etc/rdz
      CLASSPATH=.:$CFG_BASE:$CLASSPATH
      # --
$ oedit /etc/rdz/ssl/ssl.properties
  -> change: change as needed
```

The commands above copy the Developer for System z configuration files that require changes to a newly created ssl directory. The daemon.log variable in rsed.envvars must be updated to define a new log location (which is created automatically if it does not exist). The CLASSPATH update ensures that RSE can find the configuration files that were not copied to ssl. The ssl.properties file itself can be updated to match your needs.

What is left now is creating a new started task for RSE that uses a new port number and the new /etc/rdz/ssl configuration files.

Refer to the related sections in this publication for more information on the actions shown above.

All other situations

When code changes are involved (maintenance, technical previews, new release), or your changes are fairly complex, you should 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:

- SMP/E CSI
- Installation libraries
- JES Job Monitor TCP/IP port, and thus its configuration file FEJJCENFG
- JES Job Monitor startup JCL
- APPC transaction name
- RSE configuration files, rsed.envvars, *.properties and *.settings
- RSE TCP/IP port
- RSE startup JCL

A more detailed overview is listed as follows:

- SMP/E CSI
 1. Install each instance of Developer for System z in a separate CSI. SMP/E will prevent a second install of the same FMID in a CSI, but will accept installing another FMID. If the second FMID is a newer version, it will delete the existing version of the product. If the second FMID is an older version, the install will fail due to duplicate part names.
- 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/rdz. A valid sample would be /service/usr/lpp/rdz.
 2. Customization setup job FEK.SFEKSAMP(FEKSETUP) creates the data sets and directories used to store configuration files. Since the configuration files must be unique, and to avoid overwriting existing customizations, you must use unique data set and directory names when submitting this job.
- Mandatory parts
 1. JES Job Monitor configuration file FEK.#CUST.PARMLIB(FEJJCENFG) 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 FEK.#CUST.PROCLIB(JMON) refers to FEJJCENFG and therefore cannot be shared either. After renaming the member (and the JOB card if you start it as a user job) you can place all JCL's in the same data set.
 3. The RSE configuration file /etc/rdz/rsed.envvars holds references to the install path, and optionally to the server log location, which requires it to be unique. The file name is mandatory, so you cannot keep the different copies in the same directory.
 4. The ISPF.conf configuration file has a reference to FEK.SFEKPROC(FEKFRSRV), the TSO Commands server. This is software level specific, so you must create an ISPF.conf file per instance.
 5. All other z/OS UNIX based configuration files (such as *.properties) must reside in the same directory as rsed.envvars and thus cannot be shared, since rsed.envvars must be in an unshared location.
 6. The RSE startup JCL FEK.#CUST.PROCLIB(RSED) cannot be shared since it defines the TCP/IP port number and it has a reference to the install and configuration directories, which must be unique. After renaming the member (and the JOB card if you start it as a user job) you can place all JCL's in the same data set.

7. The lock daemon startup JCL FEK.#CUST.PROCLIB(LOCKD) cannot be shared since it has a reference to the installation and configuration directories, which must be unique. After renaming the member (and the JOB card if you start it as a user job) you can place all JCLs in the same data set.
- Optional parts
 1. The REXEC and SSH TCP/IP ports can be shared without any restrictions.
 2. The APPC transaction has a reference to FEK.SFEKPROC(FEKFRSRV), the TSO Commands server. This is software level specific, so you must create an APPC transaction per instance. Keep in mind that, since the APPC transaction name changes, the _FEKFSCMD_TP_NAME_ variable must be defined in rsed.envvars.
 3. Some ELAXF* procedures have a reference to hlq.SFEKLOAD, the Developer for System z load library. See the note on JCLLIB in “ELAXF* remote build procedures” on page 22 for a possible solution on making different sets available to the users.
 4. 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) DB2 stored procedure” on page 81, 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.
 5. Bidi support in CICS regions 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's name has changed.
 6. The Application Deployment Manager load modules that are included in CICS regions are backwards compatible, and thus the most recent version can be shared across releases.
 7. The Application Deployment Manager CRD VSAM is backwards compatible, and thus the most recent version can be shared across releases.

8. The Application Deployment Manager CICS resource definitions are backwards compatible, and thus the most recent version can be shared across releases.
9. CARMA VSAMs could change between software levels, thus it is not advised to share these.

Chapter 18. Migration guide

Migration considerations

This section highlights installation and configuration changes compared to previous releases of the product. It also gives some general guidelines to migrate to this release. Refer to the related sections in this manual for more information.

- 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 IBM WebSphere® Studio Enterprise Developer, it is recommended that you save the related customized files BEFORE installing the upgrade to this version of IBM Rational Developer for System z Version 7.6.1.
- Read Chapter 17, “Running multiple instances,” on page 249 if you plan on running multiple instances of Developer for System z.

Note: The migration information listed here is for Developer for System z versions that are still supported at the time of publication.

Backing up previously configured files

If you are a previous user of Developer for System z, it is recommended that you save the related customized files before installing this version of IBM Developer for System z.

Customizable Developer for system z files can be found at the following locations:

- Version 7.5
 - FEK.SFEKSAMP, some members are copied to FEK.#CUST.* by the FEKSETUP sample job, where * equals PARMLIB, PROCLIB, JCL, CNTL, ASM and COBOL
 - FEK.SFEKSAMV
 - /usr/lpp/rdz/samples/, some files are copied to /etc/rdz/ and /etc/rdz/sclmdt/* by the FEKSETUP sample job, where * equals CONFIG/, CONFIG/PROJECT/ and CONFIG/script/
- Version 7.1
 - FEK.SFEKSAMP
 - CRA.SCRASAMP
 - /usr/lpp/wd4z/rse/lib/, customizable files are advised to be copied to /etc/wd4z/
- Version 7.0
 - FEK.SFEKSAMP
 - CRA.SCRASAMP
 - /usr/lpp/wd4z/rse/lib/, customizable files are advised to be copied to /etc/wd4z/

Previous Developer for system z setups also document changes to configuration files owned by other products.

- Version 7.5
 - SYS1.PARMLIB(ASCHPMxx)
define an APPC transaction class for the TSO Commands service
 - SYS1.PARMLIB(BPXPRMxx)

- set z/OS UNIX system defaults
 - SYS1.PARMLIB(COMMNDxx)
 - start servers at IPL time
 - SYS1.PARMLIB(LPALSTxx)
 - add FEK.SFEKLPA to LPA
 - SYS1.PARMLIB(PROGxx)
 - APF authorize FEK.SFEKAUTH
 - add FEK.SFEKAUTH and FEK.SFEKLOAD to LINKLIST
 - (APPC)
 - define an APPC transaction for the TSO Commands service
 - (WLM)
 - associate an APPC transaction program with a TSO performance group
 - (WLM)
 - assign an application environment for a DB2 stored procedure
- Version 7.1
 - SYS1.PARMLIB(ASCHPMxx)
 - define an APPC transaction class for the TSO Commands service
 - SYS1.PARMLIB(BPXPRMxx)
 - set z/OS UNIX system defaults
 - SYS1.PARMLIB(PROGxx)
 - APF authorize FEK.SFEKLOAD
 - /etc/services
 - define the RSE daemon port
 - /etc/inetd.conf
 - define the RSE daemon service
 - /etc/SCLMDT/CONFIG/ISPF.conf
 - define the location of the TSO Commands server
 - (APPC)
 - define an APPC transaction for the TSO Commands service
 - (WLM)
 - associate an APPC transaction program with a TSO performance group
 - (WLM)
 - assign an application environment for a DB2 stored procedure
- Version 7.0
 - SYS1.PARMLIB(ASCHPMxx)
 - define an APPC transaction class for the TSO Commands service
 - SYS1.PARMLIB(BPXPRMxx)
 - set z/OS UNIX system defaults
 - SYS1.PARMLIB(PROGxx)
 - APF authorize FEK.SFEKLOAD
 - /etc/services
 - define the RSE daemon port
 - /etc/inetd.conf
 - define the RSE daemon service
 - (APPC)

- define an APPC transaction for the TSO Commands service
- (WLM)
- associate an APPC transaction program with a TSO performance group
- (WLM)
- assign an application environment for a DB2 stored procedure

Version 7.6.1 migration notes

The following migration notes are version 7.6.1 specific. They are valid for migrating from version 7.6, or are additions to the existing version 7.6 migration notes.

- Application Deployment Manager - Existing ADN* modules in the CICS RPL concatenation must be updated.
- Application Deployment Manager - The following sample members have been updated to add URIMAP support in the Administrative utility:
 - ADNJSPAU
 - ADNVCRD
- Application Deployment Manager - An existing CRD repository VSAM must be replaced to enable URIMAP support.
- CARMA - Added support for a variable-length layout for the CARMA custom information VSAM data set, CRASTRS
- CARMA - New sample members have been added:
 - CRA#VS2 - migrate CRASTRS to variable-length format
- JES Job Monitor - Usage of _CEE_ENVFILE_S in the started task JCL.
- JES Job Monitor - The following FEJJCNFG directives became optional:
 - HOST_CODEPAGE
- PROCLIB - New PROCLIB members have been added
 - ELAXFDCL
- RSE - Usage of 64-bit Java is now supported.
- RSE - New operator commands have been added (since version 7.6.1.0):
 - MODIFY DISPLAY PROCESS,DETAIL
- RSE - The following non-customizable directives have changed or are new in rsed.envvars (since version 7.6.0.0):
 - (_RSE_JAVAOPTS) -DDSTORE_KEEPALIVE_RESPONSE_TIMEOUT
 - (_RSE_JAVAOPTS) -DDSTORE_IO_SOCKET_READ_TIMEOUT
 - (_RSE_JAVAOPTS) -DRSECOMM_LOGFILE_MAX
- RSE - New optional directives have been added to rsed.envvars (since version 7.6.0.0 and 7.6.0.1):
 - (_RSE_JAVAOPTS) -Denable.automount
 - (_RSE_JAVAOPTS) -Ddeny.nozero.port
 - (_RSE_JAVAOPTS) -Dsingle.logon
 - (_RSE_JAVAOPTS) -Dprocess.cleanup.interval
- RSE - The following console messages have changed or are new (since version 7.6.0.1 and 7.6.1.0):
 - FEK001I
 - FEK210I

Migrate from version 7.5 to version 7.6

IBM Rational Developer for System z, FMID HHOP760

- The default SMP/E install location for MVS and z/OS UNIX components did not change and thus remain FEK.* and /usr/lpp/rdz/*.
- Application Deployment Manager - Existing ADN* modules in the CICS RPL concatenation must be updated.
- Application Deployment Manager - New load modules, which must be part of the CICS RPL concatenation, have been added to support the CICS RESTful interface.
 - ADNANAL
 - ADNCRD41
 - ADNREST
- Application Deployment Manager - New sample members have been added to support the CICS RESTful interface.
 - ADNCSDRS
 - ADNCSDTX
 - ADNTXNC
- Application Deployment Manager - Existing sample members are renamed.
 - ADNARCSD -> ADNCSDAR
 - ADNCMSGH -> ADNMSGHC
 - ADNMFEST -> ADNVMFST
 - ADNPCCSD -> ADNCSDWS
 - ADNSMSGH -> ADNMSGHS
 - ADNVSAM -> ADNVCRD
- A new, production type, RAM is provided to access CA Endevor® SCM.
 - CRARNDVR
- CARMA - New sample members have been provided to support the CA Endevor® SCM RAM.
 - FEK.#CUST.JCL(CRA#VCAD)
 - FEK.#CUST.JCL(CRA#VCAS)
 - FEK.#CUST.CNTL(CRASUBCA)
 - FEK.#CUST.PARMLIB(CRASHOW)
 - FEK.#CUST.PARMLIB(CRATMAP)
 - FEK.SFEKPROC(CRANDVRA)
 - /etc/rdz/crstart.endevor.conf
- CARMA - New sample members have been provided to support merging RAM definitions.
 - CRA#UADD
 - CRA#UQRY
- File Manager Integration - The batch interface to access File Manager is no longer supported.
- File Manager Integration - The FMEXT.properties configuration file has changed completely and must be replaced.
- JES Job Monitor - LE options are embedded in the FEJJMON load module (since version 7.5.0.1), which might require changes to your started task definition. See the FEK.SFEKSAMP(FEJJJCL) sample JCL for more details.

- JES Job Monitor - New optional directives have been added to FEJCNFG (in version 7.5.0.1 and 7.5.1.0).
 - APPLID
 - CONSOLE_NAME
 - GEN_CONSOLE_NAME
- JES Job Monitor - A new command, Show JCL, is supported (since version 7.5.1.0) which might require updates to your security software.
- Lock daemon – The lock daemon (LOCKD) is a new started task (since version 7.5.0.1). This started task can be queried to identify which Developer for z client is holding a data set lock. (System commands stop at address space level, which is the RSE thread pool.)
- SCLMDT - the default location for the SCLMDT project configuration files has changed.
 - /var/rdz/sclmdt
- RSE - New operator commands have been added.
 - MODIFY RSESTANDARDLOG
- RSE - New required directives have been added to rsed.envvars (in version 7.5.0.1 and 7.6.0.0).
 - _RSE_LOCKD_PORT
 - (_RSE_JAVAOPTS) -Dlock.daemon.port
 - (_RSE_JAVAOPTS) -Dlock.daemon.cleanup.interval
 - _RSE_LOCKD_CLASS
 - _RSE_HOST_CODEPAGE
 - (_RSE_JAVAOPTS) -Dfile.encoding
 - (_RSE_JAVAOPTS) -Dconsole.encoding
- RSE - New optional directives have been added to rsed.envvars (since version 7.5.0.1, 7.5.1.0 and 7.6.0.0).
 - (_RSE_JAVAOPTS) -Duser.log
 - (_RSE_JAVAOPTS) -Dkeep.last.log
 - (_RSE_JAVAOPTS) -Denable.standard.log
 - (_RSE_JAVAOPTS) -DDSTORE_LOG_DIRECTORY
 - (_RSE_JAVAOPTS) -DHIDE_ZOS_UNIX
 - (_RSE_JAVAOPTS) -Denable.certificate.mapping
 - GSK_CRL_SECURITY_LEVEL
 - GSK_LDAP_SERVER
 - GSK_LDAP_PORT
 - GSK_LDAP_USER
 - GSK_LDAP_PASSWORD
- RSE - Some optional directives have changed in rsed.envvars.
 - (_RSE_JAVAOPTS) -Ddaemon.log
 - (_RSE_JAVAOPTS) -Xmx
 - SCLMDT_CONF_HOME
- RSE - New optional directives have been added to ssl.properties (since version 7.5.1.0 and 7.6.0.0).
 - server_keystore_label
 - server_keystore_type

- RSE - RSE daemon supports X.509 client certificate authentication (since version 7.5.1.0), which requires updates to your current certificate and security setup when used.
- RSE - Security has been tightened, failing connection requests upon PassTicket and FEKAPPL errors.
- RSE - The default location for all log files (daemon and user logs) has changed.
 - /var/rdz/logs
 - /var/rdz/logs/\$LOGNAME
- RSE - A new sample JCL has been provided to gather Developer for System z logs and configuration information.
 - FEKLOGS

Configurable files

Table 47 gives an overview of files that are customized in version 7.6. Note that the Developer for System z sample libraries, FEK.SFEKSAMP, FEK.SFEKSAMV and /usr/lpp/rdz/samples/, come with more customizable members than listed here, such as sample CARMA source code and jobs to compile them.

Note: Sample job FEKSETUP copies all listed members to different data sets and directories, default FEK.#CUST.* and /etc/rdz/*.

Table 47. Version 7.6 customizations

Member/File	Default location	Purpose	Migration notes
FEKSETUP	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to create data sets and directories, and populate them with customizable files	Updated to include new customizable members
JMON	FEK.SFEKSAMP(FEJJJCL) [FEK.#CUST.PROCLIB]	JCL for JES Job Monitor	Added option to change LE options
FEJJJCL	FEK.SFEKSAMP [FEK.#CUST.PROCLIB(JMON)]	Shipping name for JMON member	See JMON member
RSED	FEK.SFEKSAMP(FEKRSED) [FEK.#CUST.PROCLIB]	JCL for RSE daemon	none
FEKRSED	FEK.SFEKSAMP [FEK.#CUST.PROCLIB(RSED)]	Shipping name for RSED member	See RSED member
LOCKD	FEK.SFEKSAMP(FEKLOCKD) [FEK.#CUST.PROCLIB]	JCL for lock daemon	NEW, customization is required
FEKLOCKD	FEK.SFEKSAMP [FEK.#CUST.PROCLIB(LOCKD)]	Shipping name for LOCKD member	See LOCKD member
ELAXF*	FEK.SFEKSAMP [FEK.#CUST.PROCLIB]	JCL for remote project builds, and so forth	none
FEKRACF	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL for security definitions	Minor updates
FEJJCNFG	FEK.SFEKSAMP [FEK.#CUST.PARMLIB]	JES Job Monitor configuration file	New optional directives have been added
FEJTSO	FEK.SFEKSAMP [FEK.#CUST.CNTL]	JCL for TSO submits	none
CRA\$VMSG	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to create the CARMA message VSAM	none

Table 47. Version 7.6 customizations (continued)

Member/File	Default location	Purpose	Migration notes
CRA\$VDEF	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to create the CARMA configuration VSAM	none
CRA\$VSTR	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to create the CARMA custom information VSAM	none
CRASUBMT	FEK.SFEKSAMP [FEK.#CUST.CNTL]	CARMA batch startup CLIST	none
CRASUBCA	FEK.SFEKSAMP [FEK.#CUST.CNTL]	CARMA batch startup CLIST for CA Endeavor® SCM RAM	NEW, customization is optional
CRASHOW	FEK.SFEKSAMP [FEK.#CUST.PARMLIB]	CARMA configuration for CA Endeavor® SCM RAM	NEW, customization is optional
CRATMAP	FEK.SFEKSAMP [FEK.#CUST.PARMLIB]	CARMA configuration for CA Endeavor® SCM RAM	NEW, customization is optional
CRANDVRA	FEK.SFEKPROC	CARMA allocation REXX for CA Endeavor® SCM RAM	NEW, customization is optional
CRAISPRX	FEK.SFEKSAMP [FEK.#CUST.CNTL]	Sample DD allocation exec for CARMA using TSO/ISPF Client Gateway	none
CRA#VSLM	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to create the SCLM RAM's message VSAM	none
CRA#ASLM	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to create the SCLM RAM's data sets	none
CRA#VPDS	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to create the PDS RAM's message VSAM	none
CRA#CRAM	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to compile the skeleton RAM	none
CRA#VCAD	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to create the CARMA configuration VSAM for CA Endeavor® SCM RAM	NEW, customization is optional
CRA#VCAS	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to create the CARMA custom information VSAM for CA Endeavor® SCM RAM	NEW, customization is optional
CRA#UADD	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to merge RAM definitions	NEW, customization is optional
CRA#UQRY	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to extract RAM definitions	NEW, customization is optional
CRAXJCL	FEK.SFEKSAMP [FEK.#CUST.ASM]	Sample source code for IRXJCL replacement	none
CRA#CIRX	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to compile CRAXJCL	none
ADNCSDRS	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to define the RESTful CRD server to primary CICS region	NEW, customization is optional

Table 47. Version 7.6 customizations (continued)

Member/File	Default location	Purpose	Migration notes
ADNCSDTX	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to define alternate transaction IDs to CICS region	NEW, customization is optional
ADNTXNC	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to create alternate transaction IDs	NEW, customization is optional
ADNMSGHC	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to compile ADNMSGHS	Renamed, was ADNCMSGH
ADNMSGHS	FEK.SFEKSAMP [FEK.#CUST.COBOLE]	Sample source code for the Pipeline Message Handler	Renamed, was ADNSMSGH
ADNVCRD	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to create the CRD repository	Renamed, was ADNVSAM
ADNCSDWS	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to define the Web Service CRD server to primary CICS region	Renamed, was ADNPCCSD
ADNCSDAR	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to define the CRD server to non-primary CICS regions	Renamed, was ADNARCSO
ADNJSPAU	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to update the CRD defaults	Definitions for the RESTful service are added, customizations must be redone
ADNMFST	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to create and define the Manifest repository	Renamed, was ADNMFEST
ELAXMSAM	FEK.SFEKSAMP [FEK.#CUST.PROCLIB]	JCL procedure of the WLM address space for the PL/I and COBOL Stored Procedure Builder	none
ELAXMJCL	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to define the PL/I and COBOL Stored Procedure Builder to DB2	none
FEKAPPC	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to create an APPC transaction	none
FEKAPPCL	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to display an APPC transaction	none
FEKAPPCX	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to delete an APPC transaction	none
FEKLOGS	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to collect log files	NEW, customization is optional
rsed.envvars	/usr/lpp/rdz/samples/ [etc/rdz/]	RSE environment variables	Older copies must be replaced by this one (customizations must be redone).
ISPF.conf	/usr/lpp/rdz/samples/ [etc/rdz/]	TSO/ISPF Client Gateway configuration file	ISP.SISPLIB added to SYSPROC for SCLMDT

Table 47. Version 7.6 customizations (continued)

Member/File	Default location	Purpose	Migration notes
CRASRV.properties	/usr/lpp/rdz/samples/ [/etc/rdz/]	CARMA configuration file	none
crastart.conf	/usr/lpp/rdz/samples/ [/etc/rdz/]	CARMA configuration file for CRASTART usage	none
crastart.endevor.conf	/usr/lpp/rdz/samples/ [/etc/rdz/]	CARMA configuration file for CRASTART usage for CA Endevor® SCM RAM	NEW, customization is optional
ssl.properties	/usr/lpp/rdz/samples/ [/etc/rdz/]	RSE SSL configuration file	New optional directives have been added
rsecomm.properties	/usr/lpp/rdz/samples/ [/etc/rdz/]	RSE trace configuration file	none
propertiescfg.properties	/usr/lpp/rdz/samples/ [/etc/rdz/]	Host based property groups configuration file	none
projectcfg.properties	/usr/lpp/rdz/samples/ [/etc/rdz/]	Host based projects configuration file	none
FMIEXT.properties	/usr/lpp/rdz/samples/ [/etc/rdz/]	File Manager Integration configuration file	Older copies must be replaced by this one (customizations must be redone).
uchars.settings	/usr/lpp/rdz/samples/ [/etc/rdz/]	Uneditable characters configuration file	none

Migrate from version 7.1 to version 7.5

IBM Rational Developer for System z, FMID HHOP750

- The default SMP/E install location for MVS components did not change and thus remains FEK.*.
- The default SMP/E install location for z/OS UNIX components changed to /usr/lpp/rdz/*.
- Common Access Repository Manager (CARMA) has merged into Developer for System z Version 7.5, disabling the need to install it as a separate product.
- SCLM Developer Toolkit has merged into Developer for System z Version 7.5, disabling the need to install it as a separate product.
- In version 7.5 ISPF's TSO/ISPF Client Gateway service replaces the SCLM Developer Toolkit function used in version 7.1 to connect to the TSO Commands service. The APPC connection method is still supported.
- In version 7.5, RSE server is no longer an INETD managed process but a started task. RSE server now also uses a single server model whereas with previous versions, each client-host connection had a private RSE server.
- All modules requiring APF authorization (JES Job Monitor and SCLM Developer Toolkit) moved to FEK.SFEKAUTH in version 7.5, requiring an update to the existing APF definitions.
- JES Job Monitor load module moved to FEK.SFEKAUTH in version 7.5, requiring an update to the existing started task procedure.
- CARMA load modules moved to new libraries, requiring an update to the existing CRASUBMT server startup script.

- SCLM Developer Toolkit load modules moved to new libraries, requiring and update to the existing LINKLIST definitions.
- ELAXFTS0 is a new sample build procedure since version 7.1.1, ELAXFCP1 and ELAXFPP1 are new in version 7.5.
- uchars.settings is a new configuration file for uneditable characters.
- propertiescfg.properties is a new configuration file for default property groups.
- FEJJCNFG, CRASRV.properties and FMIEXT.properties have new optional directives.
- rsed.envvars has changed in version 7.5 and must be replaced.
- The sample ISPF.conf file shipped with version 7.5 is similar to the one used by SCLM Developer Toolkit in version 7.1.
- Some of the existing Application Deployment manager customizations must be redone.
- Application Deployment Manager has new functions which require customization.
- The security settings for RSE server changed drastically in version 7.5.
- The MVS.MCSOPER.JMON security profile is new for JES Job monitor in version 7.5.
- The CARMA startup script changed name and moved to a new location, requiring an update to the existing CRASRV.properties configuration file.
- The FMI startup script changed name and moved to a new location, requiring an update to the existing FMIEXT.properties configuration file.
- A new load module has been added for bidirectional support, requiring an update to the existing CICS DFHRPL concatenation if you are not using the FEK.SFEKLOAD library.
- Changes to the MAXPROCUSER parameter of SYS1.PARMLIB(BPXPRMxx) are now also documented.

Configurable files

Table 48 gives an overview of files that are customized in version 7.5. Note that the Developer for System z sample libraries, FEK.SFEKSAMP, FEK.SFEKSAMV and /usr/lpp/rdz/samples/, come with more customizable members than listed here, such as sample CARMA source code and jobs to compile them.

Note: Sample job FEKSETUP copies all listed members to different data sets and directories, default FEK.#CUST.* and /etc/rdz/*.

Table 48. Version 7.5 customizations

Member/File	Default location	Purpose	Migration notes
FEKSETUP	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to create data sets and directories, and populate them with customizable files	NEW, customization is required
JMON	FEK.SFEKSAMP(FEJJJCL) [FEK.#CUST.PROCLIB]	JCL for JES Job Monitor	STEPLIB changed to SFEKAUTH
RSED	FEK.SFEKSAMP(FEKRSED) [FEK.#CUST.PROCLIB]	JCL for RSE daemon	NEW, customization is required

Table 48. Version 7.5 customizations (continued)

Member/File	Default location	Purpose	Migration notes
ELAXF*	FEK.SFEKSAMP [FEK.#CUST.PROCLIB]	JCL for remote project builds, and so on	ELAXFTSO, ELAXFCP1 and ELAXFPP1 are new
FEKRACF	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL for security definitions	NEW, required
FEJCNFG	FEK.SFEKSAMP [FEK.#CUST.PARMLIB]	JES Job Monitor configuration file	<ul style="list-style-type: none"> Some directives became optional New optional directives have been added
FEJTSO	FEK.SFEKSAMP [FEK.#CUST.CNTL]	JCL for TSO submits	Job name can now be a variable
CRAISPRX	FEK.SFEKSAMP [FEK.#CUST.CNTL]	Sample DD allocation exec for CARMA using TSO/ISPF Client Gateway	NEW, customization is optional
CRAJCL	FEK.SFEKSAMP [FEK.#CUST.ASM]	Sample source code for IRXJCL replacement	NEW, customization is optional
CRA#CIRX	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to compile CRAJCL	NEW, customization is optional
ADNSMSGH	FEK.SFEKSAMP [FEK.#CUST.COBOLE]	Sample source code for the Pipeline Message Handler	Older copies must be replaced by this one (customizations must be redone)
ADNPCCSD	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to define the CRD server to primary CICS region	Older copies must be replaced by this one (customizations must be redone)
ADNJSPAU	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to update the CRD defaults	NEW, customization is optional
ADNMFEST	FEK.SFEKSAMP [FEK.#CUST.JCL]	JCL to create and define the Manifest repository	NEW, customization is optional
rsed.envvars	/usr/lpp/rdz/samples/ [etc/rdz/]	RSE environment variables	Older copies must be replaced by this one (customizations must be redone)
ISPF.conf	/usr/lpp/rdz/samples/ [etc/rdz/]	TSO/ISPF Client Gateway configuration file	Identical to the ISPF.conf shipped with SCLMDT in v7.1

Table 48. Version 7.5 customizations (continued)

Member/File	Default location	Purpose	Migration notes
CRASRV.properties	/usr/lpp/rdz/samples/ [/etc/rdz/]	CARMA configuration file	<ul style="list-style-type: none"> Startup script changed location and name New optional directives have been added
crastart.conf	/usr/lpp/rdz/samples/ [/etc/rdz/]	CARMA configuration file for CRASTART usage	NEW, customization is optional
FMEXT.properties	/usr/lpp/rdz/samples/ [/etc/rdz/]	File Manager Integration configuration file	<ul style="list-style-type: none"> Startup script changed location and name New optional directives have been added
uchars.settings	/usr/lpp/rdz/samples/ [/etc/rdz/]	Uneditable characters configuration file	NEW, customization is optional

Migrate from version 7.0 to version 7.1

IBM Rational Developer for System z, FMID HHOP710

- The default SMP/E install location for MVS and z/OS UNIX components did not change and thus remains FEK.* and /usr/lpp/wd4z/*.
- Added:** Setup choice - TSO/ISPF commands through an APPC transaction or through 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/FMEXT.properties
- Changed:** The following members have moved:
 - SFEKDLL(FEJBDTRX) -> SFEKLOAD(FEJBDTRX)
- Changed:** The following customizable members have changed:
 - samplib FEKFAPPC
 - /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

- The default SMP/E install location for MVS components did not change and thus remains CRA.*.
- 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\$VMSG
 - 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

Configurable files

Table 23 gives an overview of files that are customized in version 7.1. Note that the CARMA and Developer for System z sample libraries, CRA.SCRASAMP, FEK.SFEKSAMP and /usr/lpp/wd4z/rse/lib/, come with more customizable members than listed here, such as sample CARMA source code and jobs to compile them.

Table 49. Version 7.1 customizations

Member/File	Default location	Purpose	Migration notes
ELAXF*	FEK.SFEKSAMP	JCL for remote project builds, and other jobs	ELAXFADT is new
CRA\$VMSG	CRA.SCRASAMP	JCL to create the CARMA message VSAM	<ul style="list-style-type: none"> • renamed, was CRAMREPR • the VSAM created by this job is updated
CRA\$VDEF	CRA.SCRASAMP	JCL to create the CARMA configuration VSAM	<ul style="list-style-type: none"> • renamed, was CRAREPR • the VSAM created by this job is updated
CRA\$VSTR	CRA.SCRASAMP	JCL to create the CARMA custom information VSAM	renamed, was CRASREPR
CRASUBMT	CRA.SCRASAMP	CARMA batch startup CLIST	add DD CARMALOG

Table 49. Version 7.1 customizations (continued)

Member/File	Default location	Purpose	Migration notes
CRA#VSLM	CRA.SCRASAMP	JCL to create the SCLM RAM's message VSAM	renamed, was CRALREPR
CRA#ASLM	CRA.SCRASAMP	JCL to create the SCLM RAM's data sets	renamed, was CRASALX
CRA#VPDS	CRA.SCRASAMP	JCL to create the PDS RAM's message VSAM	renamed, was CRATREPR
CRA#CRAM	CRA.SCRASAMP	JCL to compile the skeleton RAM	renamed, was CRARAMCM
FEKAPPCC	FEK.SFEKSAMP	JCL to create an APPC transaction	exploit ISPF NEST support
rsed.envvars	/usr/lpp/wd4z/rse/lib/ [/etc/wd4z/]	RSE environment variables	Older copies must be replaced by this one (customizations must be redone)
FMIEXT.properties	/usr/lpp/wd4z/rse/lib/ [/etc/wd4z/]	File Manager Integration configuration file	NEW, customization is required when used

Appendix A. Setting up SSL and X.509 authentication

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. This appendix also provides a sample setup to support users authenticating themselves with an X.509 certificate.

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. 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 *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 RSE daemon connection that will use SSL. We will also create our own security certificates to be used by the different parts of the RSE connection.

- "Decide where to store private keys and certificates"
- "Create a key ring with RACF" on page 271
- "Clone the existing RSE setup" on page 272
- "Update rsed.envvars to enable coexistence" on page 273
- "Update ssl.properties to enable SSL" on page 273
- "Activate SSL by creating a new RSE daemon" on page 273
- "Test the connection" on page 274
- "(Optional) Add X.509 client authentication support" on page 277
- "(Optional) Create a key database with gskkyman" on page 277
- "(Optional) Create a key store with keytool" on page 280

Throughout this appendix, a uniform naming convention is used:

- Certificate : rdzrse
- Key and certificate storage : rdzssl.*
- Password : rsessl
- Daemon user ID : stcrse

Some 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.

Decide where to store private keys and certificates

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.

However, all implementations follow the same principle. A command generates a key pair (a public key and associated private key). The command then wraps the public key into an X.509 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 key file.

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 SAF-compliant security software (for example, RACF). The RSE server (which is started by the daemon) is a Java SSL application and uses a key store file created by keytool or a key ring managed by your security software.

Table 50. SSL certificate storage mechanisms

Certificate storage	Created and managed by	RSE daemon	RSE server
key ring	SAF-compliant security product	supported	supported
key database	z/OS UNIX's gskkyman	supported	/
key store	Java's keytool	/	supported

To connect through SSL, we need both the key store and the key database, either as a z/OS UNIX file or as a SAF-compliant key ring:

- key store (RACF or keytool)
- key database (RACF or gskkyman)

Note:

- SAF-compliant key rings are the preferred method for managing certificates.
- A shared certificate can be used if RSE daemon and RSE server use the same certificate management method.
- RSE daemon must run program controlled. Using System SSL within implies that SYS1.SIEALNKE must be made program controlled by your security software.
- 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.
STEPLIB=\$STEPLIB:SYS1.SIEALNKE
Be aware, however, that:
 - Using STEPLIB in z/OS UNIX has a negative performance impact.
 - If one STEPLIB library is APF authorized, then all must be authorized. Libraries lose their APF authorization when they are mixed with non-authorized libraries in STEPLIB.
- 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. Refer to *System SSL Programming* (SC24-5901) for more information.

Refer to *Security Server RACF Security Administrator's Guide* (SA22-7683) for information on RACF and digital certificates. gskkyman documentation can be found in *System SSL Programming* (SC24-5901), and keytool documentation is available at <http://java.sun.com/j2se/1.5.0/docs/tooldocs/solaris/keytool.html>.

Create a key ring with RACF

Do not execute this step if you use gskkyman to create the RSE daemon key database and keytool to create the RSE server key store.

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 *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(stcrse)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ACCESS(READ) ID(stcrse)
SETROPTS RACLIST(FACILITY) REFRESH
```

```
RACDCERT ID(stcrse) GENCERT SUBJECTSDN(CN('rdz rse ssl') +
OU('rdz') O('IBM') L('Raleigh') SP('NC') C('US')) +
NOTAFTER(DATE(2017-05-21)) WITHLABEL('rdzrse') KEYUSAGE(HANDSHAKE)
```

```
RACDCERT ID(stcrse) ADDRING(rdzssl.racf)
RACDCERT ID(stcrse) CONNECT(LABEL('rdzrse') RING(rdzssl.racf) +
DEFAULT USAGE(PERSONAL))
```

The sample above starts by creating the necessary profiles and permitting user ID STCRSE access to key rings and certificates owned by that user ID. The user ID used must match the user ID used to run the SSL RSE daemon. The next step is creating a new, self-signed, certificate with label rdzrse. No password is needed. This certificate is then added to a newly created key ring (rdzssl.racf). Just as with the certificate, no password is needed for the key ring.

The result can be verified with the following list option:

```
RACDCERT ID(stcrse) LIST
Digital certificate information for user STCRSE:

Label: rdzrse
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=rdz rse ssl.OU=rdz.O=IBM.L=Raleigh.SP=NC.C=US<
Subject's Name:
>CN=rdz rse ssl.OU=rdz.O=IBM.L=Raleigh.SP=NC.C=US<
Private Key Type: Non-ICSF
Private Key Size: 1024
Ring Associations:
Ring Owner: STCRSE
Ring:
>rdzssl.racf<
```

(Optional) Using a signed certificate

Certificates can be either self-signed or signed by a Certificate Authority (CA). A certificate signed by a CA means that the CA guarantees that the owner of the certificate is who he claims to be. The signing process adds the CA credentials (also a certificate) to your certificate, making it a multi-element certificate chain.

When using a certificate signed by a CA you can avoid trust validation questions by the Developer for System z client, if the client already trusts the CA.

Follow these steps to create and use a CA signed certificate:

1. Create a self-signed certificate.
`RACDCERT ID(stcrse) GENCERT WITHLABEL('rdzrse') . . .`
 2. Create a signing request for this certificate.
`RACDCERT ID(stcrse) GENREQ (LABEL('rdzrse')) DSN(dsn)`
 3. Send the signing request to your CA of choice.
 4. Check if the CA credentials (also a certificate) are already known.
`RACDCERT CERTAUTH LIST`
 5. Mark the CA certificate as trusted.
`RACDCERT CERTAUTH ALTER(LABEL('CA cert')) TRUST`
Or add the CA certificate to the database.
`RACDCERT CERTAUTH ADD(dsn) WITHLABEL('CA cert') TRUST`
 6. Add the signed certificate to the database; this will replace the self-signed one.
`RACDCERT ID(stcrse) ADD(dsn) WITHLABEL('rdzrse') TRUST`
- Note:** Do NOT delete the self-signed certificate before replacing it. If you do, you lose the private key that goes with the certificate, which makes the certificate useless.
7. Create a key ring.
`RACDCERT ID(stcrse) ADDRING(rdzssl.racf)`
 8. Add the signed certificate to the key ring.
`RACDCERT ID(stcrse) CONNECT(ID(stcrse) LABEL('rdzrse'))
RING(rdzssl.racf))`
 9. Add the CA certificate to the key ring.
`RACDCERT ID(stcrse) CONNECT(CERTAUTH LABEL('CA cert'))
RING(rdzssl.racf))`

Clone the existing RSE setup

In this step a new instance of the RSE configuration files is created, so that the SSL setup can run parallel with the existing one(s). The following sample commands expect the configuration files to be in `/etc/rdz/`, which is the default location used in “Customization setup” on page 13.

```
$ cd /etc/rdz
$ mkdir ssl
$ cp rsed.envvars ssl
$ cp ssl.properties ssl
$ ls ssl
rsed.envvars    ssl.properties
```

The z/OS UNIX commands listed above create a subdirectory called `ssl` and populate it with the configuration files that require changes. We can share the other configuration files, the installation directory, and the MVS components, because they are not SSL-specific.

By reusing most of the existing configuration files, we can focus on the changes that are actually required for setting up SSL and avoid doing the complete RSE setup again. (For example, we can avoid defining a new location for `ISPF.conf`.)

Update rsed.envvars to enable coexistence

So far, the definitions are an exact copy of the current setup, which implies that the logs of the new RSE daemon will overlay the current server log files. RSE also needs to know where to find the configuration files that were not copied to the ssl directory. Both issues can be addressed by minor changes to rsed.envvars.

```
$ oedit /etc/rdz/ssl/rsed.envvars
-> uncomment and change: -Ddaemon.log=/var/rdz/logs/ssl
-> add at the END:
# -- NEEDED TO FIND THE REMAINING CONFIGURATION FILES
CFG_BASE=/etc/rdz
CLASSPATH=.:$CFG_BASE:$CLASSPATH
# --
```

The changes above define a new log location (which will be created by RSE daemon if the log location does not exist). The changes also update the CLASSPATH so that the SSL RSE processes will first search the current directory (/etc/rdz/ssl) for configuration files and then search the original directory (/etc/rdz).

Update ssl.properties to enable SSL

By updating ssl.properties, RSE is instructed to start using SSL encrypted communication.

```
$ oedit /etc/rdz/ssl/ssl.properties
-> change: enable_ssl=true
-> uncomment and change: daemon_keydb_file=rdzssl.racf
-> uncomment and change: daemon_key_label=rdzrse
-> uncomment and change: server_keystore_file=rdzssl.racf
-> uncomment and change: server_keystore_label=rdzrse
-> uncomment and change: server_keystore_type=JCERACFKS
```

The changes above enable SSL and tell the RSE daemon and RSE server that their (shared) certificate is stored under label rdzrse in key ring rdzssl.racf. The JCERACFKS keyword tells RSE server that a SAF-compliant key ring is used as key store.

Activate SSL by creating a new RSE daemon

As stated before, we will create a second connection that will use SSL, which implies creating a new RSE daemon. The RSE daemon can be a started task or user job. We will use the user job method for initial (test) setup. The following instructions expect the sample JCL to be in FEK.#CUST.PROCLIB(RSED), which is the default location used in “Customization setup” on page 13:

1. Create a new member FEK.#CUST.PROCLIB(RSEDSSL) and copy in sample JCL FEK.#CUST.PROCLIB(RSED).
2. Customize RSEDSSL by adding a job card on top and an exec statement at the bottom. Also provide a new port number (4047) and the location of the SSL-related configuration files (/etc/rdz/ssl), as shown in the following code sample. Note that we enforce the usage of user ID STCRSE, as this user ID was given the proper access authority to certificates and key rings in a previous step.

```

//RSEDSSL JOB CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1),USER=STCRSE
//*
/* RSE DAEMON - SSL
/*
//RSED      PROC IVP='',          * 'IVP' to do an IVP test
//          PORT=4047,
//          HOME='/usr/lpp/rdz',
//          CNFG='/etc/rdz/ssl'
/*
//RSE      EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT,
//          PARM='PGM &HOME./bin/rsed.sh &IVP &PORT &CNFG'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//PEND
/*
//RSED      EXEC RSED
/*

```

Figure 61. RSEDSSL - RSE daemon user job for SSL

Note: The user ID assigned to the RSEDSSL job must have the same authorizations as the original RSE daemon. FACILITY profile BPX.SERVER and PTKTDATA profile IRRPTAUTH.FEKAPPL.* are the key elements here.

Test the connection

The SSL host configuration is complete and the RSE daemon for SSL can be started by submitting job FEK.#CUST.PROCLIB(RSEDSSL), which was created earlier.

The new setup can now 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 on the client, using port 4047 for the RSE daemon.

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 or the CA that signed it, Developer for System z client will prompt the user asking if this certificate can be trusted.



Figure 62. Import Host Certificate dialog

By clicking the Finish button the user can accept this certificate as trusted, after which the connection initialization continues.

Note: RSE daemon and RSE server might use two different certificate locations, resulting in two different certificates and thus two 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:

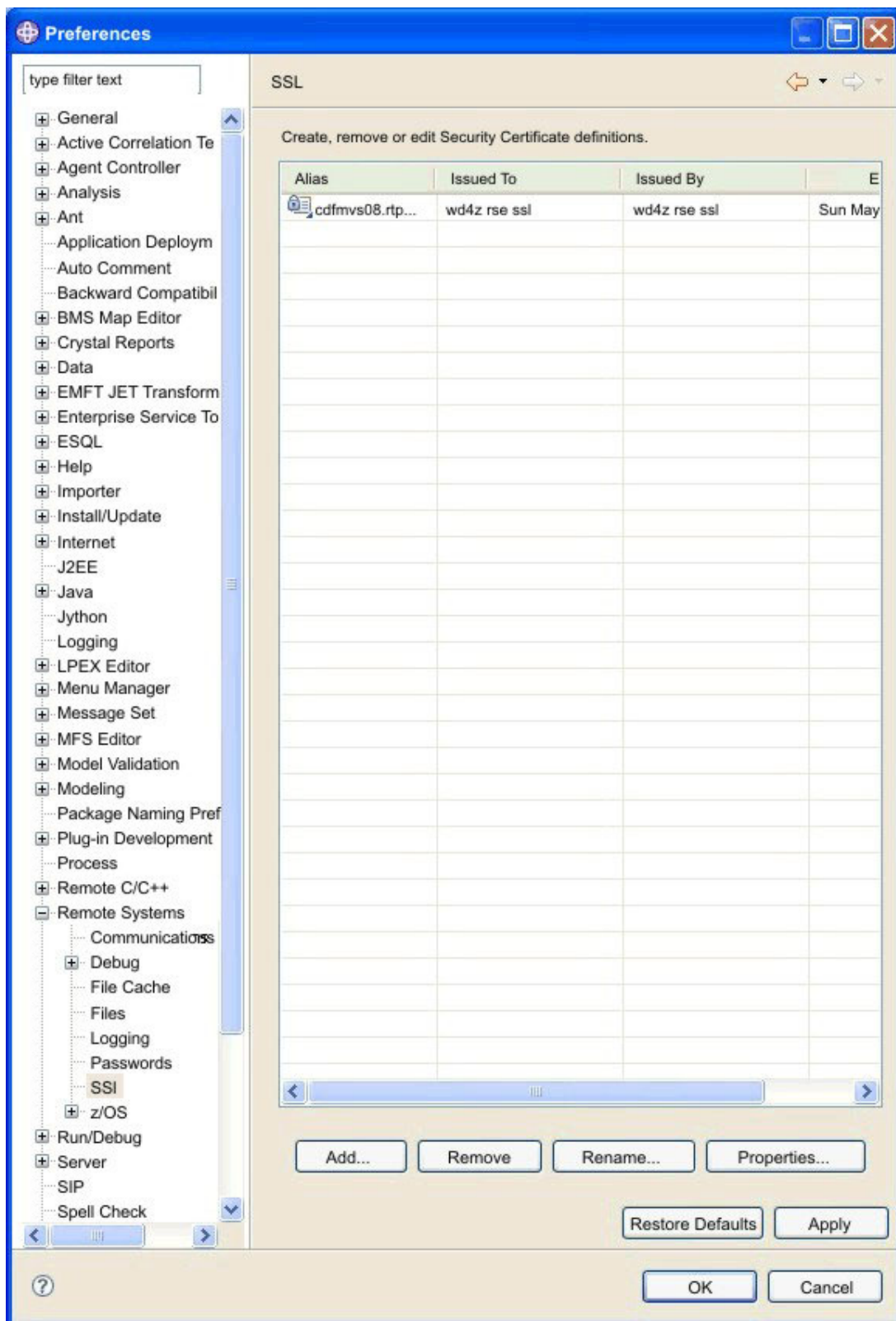


Figure 63. Preferences dialog - SSL

If SSL communication fails, the client will return an error message. More information is available in the different server and user log files, as described in “RSE daemon and thread pool logging” on page 129 and “RSE user logging” on page 130.

(Optional) Add X.509 client authentication support

RSE daemon supports users authenticating themselves with an X.509 certificate. Using SSL encrypted communication is a prerequisite for this function, because it is an extension to the host authentication with a certificate used in SSL.

There are multiple ways to do certificate authentication for a user, as described in “Client authentication using X.509 certificates” on page 158. The next steps document the setup needed to support the method where your security software authenticates the certificate using the HostIdMappings certificate extension.

1. Change the certificate that identifies the Certificate Authority (CA) used to sign the client certificate to a highly trusted CA certificate. Although the TRUST status is sufficient for certificate validation, a change to HIGHTRUST is done, because it is used for the certificate authentication part of the logon process.

```
RACDCERT CERTAUTH ALTER(LABEL('HighTrust CA')) HIGHTRUST
```

2. Add the CA certificate to the key ring, `rdzssl.racf`, so that it is available to validate the client certificates.

```
RACDCERT ID(stcrse) CONNECT(CERTAUTH LABEL('HighTrust CA') +  
RING(rdzssl.racf))
```

This concludes the security software setup for the CA certificate.

3. Define a resource (format `IRR.HOST.hostname`) in the `SERVAUTH` class for the host name, `CDFMVS08.RALEIGH.IBM.COM`, defined in the HostIdMappings extension of your client certificate.

```
RDEFINE SERVAUTH IRR.HOST.CDFMVS08.RALEIGH.IBM.COM UACC(NONE)
```

4. Grant the RSE started task user ID, `STCRSE`, access to this resource with `READ` authority.

```
PERMIT IRR.HOST.CDFMVS08.RALEIGH.IBM.COM CLASS(SERVAUTH) +  
ACCESS(READ) ID(stcrse)
```

5. Activate your changes to the `SERVAUTH` class. Use the first command if the `SERVAUTH` class is not active yet. Use the second one to refresh an active setup.

```
SETROPTS CLASSACT(SERVAUTH) RACLIST(SERVAUTH)
```

or

```
SETROPTS RACLIST(SERVAUTH) REFRESH
```

This concludes the security software setup for the HostIdMappings extension.

6. Restart the RSE started task to start accepting client logons using X.509 certificates.

(Optional) Create a key database with gskkyman

Do not execute this step if you use an SAF-compliant key ring for the RSE daemon key database.

`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`. Refer to *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

```

```

$ cd /etc/rdz/ssl
$ gskkyman          Database Menu

```

1 - Create new database

```

Enter option number: 1
Enter key database name (press ENTER to return to menu): rdzssl.kdb
Enter database password (press ENTER to return to menu): rsessl
Re-enter database password: rsessl
Enter password expiration in days (press ENTER for no expiration):
Enter database record length (press ENTER to use 2500):

```

Key database /etc/rdz/ssl/rdzssl.kdb created.

Press ENTER to continue.

Key Management Menu

6 - Create a self-signed certificate

Enter option number (press ENTER to return to previous menu): 6

Certificate Type

5 - User or server certificate with 1024-bit RSA key

```

Select certificate type (press ENTER to return to menu): 5
Enter label (press ENTER to return to menu): rdzrse
Enter subject name for certificate
  Common name (required): rdz rse ssl
  Organizational unit (optional): rdz
  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

0 - Exit program

Enter option number (press ENTER to return to previous menu): 0

```
$ ls -l rdzssl.*
```

```

total 152
-rw----- 1 IBMUSER SYS1      35080 May 24 14:24 rdzssl.kdb
-rw----- 1 IBMUSER SYS1       80 May 24 14:24 rdzssl.rdb

```

```
$ chmod 644 rdzssl.*
```

```
$ ls -l rdzssl.*
```

```

-rw-r--r-- 1 IBMUSER SYS1      35080 May 24 14:24 rdzssl.kdb
-rw-r--r-- 1 IBMUSER SYS1       80 May 24 14:24 rdzssl.rdb

```

The sample above starts by creating a key database called `rdzssl.kdb` with password `rsessl`. Once the database exists, it is populated by creating a new, self-signed, certificate, valid for about 10 years (not counting leap days). The

certificate is stored under the label `rdzrse` and with the same password (`rsessl`) as the one used for the key database (this is an 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. 644 (owner has read/write, everyone has read) is a usable mask for the `chmod` command.

The result can be verified by selecting the **Show certificate information** option in the **Manage keys and certificates** submenu, as follows:

```
$ gskkyman
```

```
Database Menu
```

```
2 - Open database
```

```
Enter option number: 2
```

```
Enter key database name (press ENTER to return to menu): rdzssl.kdb
```

```
Enter database password (press ENTER to return to menu): rsessl
```

```
Key Management Menu
```

```
1 - Manage keys and certificates
```

```
Enter option number (press ENTER to return to previous menu): 1
```

```
Key and Certificate List
```

```
1 - rdzrse
```

```
Enter label number (ENTER to return to selection menu, p for previous list): 1
```

```
Key and Certificate Menu
```

```
1 - Show certificate information
```

```
Enter option number (press ENTER to return to previous menu): 1
```

```
Certificate Information
```

```
Label: rdzrse
Record ID: 14
Issuer Record ID: 14
Trusted: Yes
Version: 3
Serial number: 45356379000ac997
Issuer name: rdz rse ssl
rdz
IBM
Raleigh
NC
US
Subject name: rdz rse ssl
rdz
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

0 - Exit program

Enter option number (press ENTER to return to previous menu): 0

The following `ssl.properties` sample shows that the `daemon_*` directives differ from the SAF key ring sample shown earlier.

```
$ oedit /etc/rdz/ssl/ssl.properties
-> change: enable_ssl=true
-> uncomment and change: daemon_keydb_file=rdzssl.kdb
-> uncomment and change: daemon_keydb_password=rsessl
-> uncomment and change: daemon_key_label=rdzrse
-> uncomment and change: server_keystore_file=rdzssl.racf
-> uncomment and change: server_keystore_label=rdzrse
-> uncomment and change: server_keystore_type=JCERACFKS
```

The changes above enable SSL and tell the RSE daemon that the certificate is stored under label `rdzrse` in key database `rdzssl.kdb` with password `rsessl`. RSE server is still using a SAF compliant key ring.

(Optional) Create a key store with keytool

Do not execute this step if you use a SAF-compliant key ring for the RSE server key store.

"`keytool -genkey`" generates a private key pair and a matching self-signed certificate, which is 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`, where `/usr/lpp/java/J5.0` is the directory where Java is installed:

```
PATH=$PATH:/usr/lpp/java/J5.0/bin
```

All information can be passed as a parameter, but due to command-line length limitations some interactivity is required, as follows:

```
$ cd /etc/rdz/ssl
$ keytool -genkey -alias rdzrse -validity 3650 -keystore rdzssl.jks -storepass
rsessl -keypass rsessl
What is your first and last name?
[Unknown]: rdz rse ssl
What is the name of your organizational unit?
[Unknown]: rdz
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=rdz rse ssl, OU=rdz, O=IBM, L=Raleigh, ST=NC, C=US correct? (type "yes"
or "no")
[no]: yes
$ ls -l rdzssl.*
-rw-r--r-- 1 IBMUSER SYS1          1224 May 24 14:17 rdzssl.jks
```

The self-signed certificate created above is valid for about 10 years (not counting leap days). It is stored in `/etc/rdz/ssl/rdzssl.jks` using alias `rdzrse`. 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, as follows:

```
$ keytool -list -alias rdzrse -keystore rdzssl.jks -storepass rsessl -v
Alias name: rdzrse
Creation date: May 24, 2007
Entry type: keyEntry
Certificate chain length: 1
Certificate 1":
Owner: CN=rdz rse ssl, OU=rdz, O=IBM, L=Raleigh, ST=NC, C=US
Issuer: CN=rdz rse ssl, OU=rdz, 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
```

The following `ssl.properties` sample shows that the `server_*` directives differ from the SAF key ring sample shown earlier.

```
$ oedit /etc/rdz/ssl/ssl.properties
-> change: enable_ssl=true
-> uncomment and change: daemon_keydb_file=rdzssl.racf
-> uncomment and change: daemon_key_label=rdzrse
-> uncomment and change: server_keystore_file=rdzssl.jks
-> uncomment and change: server_keystore_password=rsessl
-> uncomment and change: server_keystore_label=rdzrse
-> optionally uncomment and change: server_keystore_type=JKS
```

The changes above enable SSL and tell the RSE server that the certificate is stored under label `rdzrse` in key store `rdzssl.jks` with password `rsessl`. RSE daemon is still using a SAF-compliant key ring.

Appendix B. 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 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

When using APPC for the TSO Commands service, Developer for System z is dependent 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 fekfivpt Installation Verification Program (IVP). The command should return an output like that in this sample (\$ is the z/OS UNIX prompt):

```
$ fekfivpt
```

```
Wed Jul  2 13:11:54 EDT 2008
uid=1(USERID) gid=0(GROUP)
using /etc/rdz/rsed.envvars
```

```
-----
TCP/IP resolver configuration (z/OS UNIX search order):
-----
```

```
Resolver Trace Initialization Complete -> 2008/07/02 13:11:54.745964
```

```
res_init Resolver values:
```

```
Global Tcp/Ip Dataset = None
Default Tcp/Ip Dataset = None
Local Tcp/Ip Dataset  = /etc/resolv.conf
Translation Table     = Default
UserId/JobName         = USERID
Caller API             = LE C Sockets
Caller Mode            = EBCDIC
(L) DataSetPrefix     = TCPIP
(L) HostName           = CDFMVS08
(L) TcpIpJobName       = TCPIP
(L) DomainOrigin      = RALEIGH.IBM.COM
(L) NameServer         = 9.42.206.2
                      9.42.206.3
(L) NsPortAddr         = 53           (L) ResolverTimeout      = 10
(L) ResolveVia         = UDP          (L) ResolverUdpRetries   = 1
(*) Options NDots      = 1
(*) SockNoTestStor     =
(*) AlwaysWto          = NO           (L) MessageCase         = MIXED
(*) LookUp              = DNS LOCAL
```

```
res_init Succeeded
```

```
res_init Started: 2008/07/02 13:11:54.755363
```

```
res_init Ended: 2008/07/02 13:11:54.755371
```

```
*****
```

```
MVS TCP/IP NETSTAT CS V1R9      TCPIP Name: TCPIP      13:11:54
```

```
Tcpip started at 01:28:36 on 06/23/2008 with IPv6 enabled
```

```
-----
```

host IP address:

```
-----  
hostName=CDFMVS08  
hostAddr=9.42.112.75  
bindAddr=9.42.112.75  
localAddr=9.42.112.75
```

Success, addresses match

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.

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 the following:

1. **GLOBALTCPIPDATA**

If defined, the resolver GLOBALTCPIPDATA setup statement value is used (see also "Understanding resolvers" on page 284). 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, task, or 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 284).

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 the following. The search order ends at the first file being 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 the following. The search ends at the first file being 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 set up information to Developer for System z

As stated before, Developer for System z is dependent upon TCP/IP having the correct hostname when it is initialized, when using APPC. This implies that the different TCP/IP and Resolver configuration files must be set up correctly.

In the following example we will focus on some configuration tasks for TCP/IP and Resolver. Note that this does not cover a complete setup 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=EZBTCPIP,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 do not use a domain name server (DNS); all names will be resolved through 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 284, 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 51 on page 288 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 285, 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 51 also tells us that if we do not 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, you should keep the TCP/IP and Resolver configuration files in sync with each other.

Table 51. 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

Table 51. Local definitions available to resolver (continued)

File type description	APIs affected	Candidate files
Local host tables	endhostent endnetent getaddrinfo gethostbyaddr gethostbyname gethostent GetHostNumber GetHostResol GetHostString getnameinfo getnetbyaddr getnetbyname getnetent IsLocalHost Resolve sethostent setnetent	IPv4 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 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: Table 51 on page 288 is a partial copy from a table in *Communications Server: IP Configuration Guide* (SC31-8775). See that manual for the full table.

Host address is not resolved correctly

When you see problems where TCP/IP Resolver cannot resolve the host address properly, it is most likely due to a missing or incomplete resolver configuration file. A clear indication for this problem is the following message in `lock.log`:

```
clientip(0.0.0.0) <> callerip(<host IP address>)
```

To verify this, execute the `febfivpt` TCP/IP IVP, as described in Chapter 7, “Installation verification,” on page 99. The resolver configuration section of the output will look like the following sample:

```
Resolver Trace Initialization Complete -> 2008/07/02 13:11:54.745964
```

```
res_init Resolver values:
Global Tcp/Ip Dataset = None
Default Tcp/Ip Dataset = None
Local Tcp/Ip Dataset = /etc/resolv.conf
Translation Table = Default
UserId/JobName = USERID
Caller API = LE C Sockets
Caller Mode = EBCDIC
```

Ensure that the definitions in the file (or data set) referenced by “Local Tcp/Ip Dataset” are correct.

This field will be blank if you do not use a default name for the IP resolver file (using the z/OS UNIX search order). If so, add the following statement to `rsed.envvars`, where `<resolver file>` or `<resolver data>` represents the name of your IP resolver file:

```
RESOLVER_CONFIG=<resolver file>
```

or

```
RESOLVER_CONFIG='<resolver data set>'
```

Appendix C. 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 or modifying an existing setup. INETD is used by Developer for System z for REXEC/SSH functionality.

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 (such as 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 embedded 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 an 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 four 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 the following. The search ends at the first file being 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 the following. The search ends at the first data set being 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 an MVS load module, such as 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, you should 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 two 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

You should 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 sent 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 started tasks 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 the following code sample is a sample procedure to start INETD (the KILL step removes an active INETD process, if any):

```
//INETD    PROC PRM=
//*
//KILL      EXEC PGM=BPXBATCH,REGION=0M,
//          PARM='SH ps -e | grep inetd | cut -c 1-10 | xargs -n 1 kill'
//*
//INETD     EXEC PGM=BPXBATCH,REGION=0M,
//          PARM='PGM /usr/sbin/inetd &PRM'
//STDERR    DD SYSOUT=*
//* STDIN, STDOUT and STDENV are defaulted to /dev/null
//*
```

Figure 64. INETD startup JCL

Note:

- STDIN, STDOUT and STDERR must be z/OS UNIX files when allocated. STDENV can be either a MVS data set or a z/OS UNIX file. Since z/OS 1.7, SYSOUT can be assigned to STDOUT and STDERR. Refer to *UNIX System Services Command Reference* (SA22-7802) to learn more about BPXBATCH.
- 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 in 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):

```
# ps -e | grep inetd
7 ?          0:00 /usr/sbin/inetd
# kill 7
# _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 such as 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 an 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 dependent upon INETD for managing REXEC and/or SSH. It might also impose extra requirements on top of the INETD setup described above.

RExec (or SSH) is used for the following two purposes, as described in “(Optional) Using REXEC (or SSH)” on page 93.

- remote (host-based) actions in z/OS UNIX subprojects
- alternative RSE server startup method

The remote actions in z/OS UNIX subprojects do not require special settings. The alternative RSE startup method however does require special settings.

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.

REXEC (or SSH)

The REXEC (or SSH) daemon that is started by INETD when a client connects to port 512 (or 22, respectively) 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 REXEC (or SSH) 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/rdz/*`
- READ and EXECUTE access to the Developer for System z configuration directories, default `/etc/rdz/*`

Appendix D. 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 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 the following 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 following job 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 65. 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, such as 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 started task). 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 66. SYS1.SAMPLIB(ATBAPPL)

Refer to *Communications Server IP SNA Network Implementation Guide* (SC31-8777) for more information on configuring VTAM.

SYS1.PARMLIB(APPCPMxx)

To enable and support the flow of conversations between systems, sites must define Logical Units (LUs) 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.

The following code sample is a 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 67. 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 the following:

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).

Note: Operator command **D APPC,LU,ALL** will show all active LU definitions and mark the base LU.

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 “`rsed.envvars`, RSE configuration file” on page 28 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.

The TSO Commands service also needs the default specifications to be specified in the `OPTIONS` and `TPDEFAULT` sections.

The following code sample is a `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 68. 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.
- Operator command **D ASCH,ALL** will show all active APPC transaction scheduler classes.

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 is not active yet. Enter the following console commands to start APPC/MVS (where xx equals the last two 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 the following console **SET** command (where xx equals the last two 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* (GC28-1781) 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) APPC transaction for the TSO Commands service” on page 95.

The documented way to define the APPC transaction is by customizing and submitting FEK.#CUST.JCL(FEKAPPCC).

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* (SC23-5885-00) whitepaper is available at the Developer for System z internet library, <http://www-306.ibm.com/software/awdtools/rdz/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'
```

(Optional) Alternative setup options

Developer for System z supports alternative APPC and VTAM setup options, some of which are documented in this section.

Alternative transaction name

The default transaction name for the TSO Commands service is FEKFRSRV, as described in “(Optional) APPC transaction for the TSO Commands service” on page 95. As described in the same section, this name can be changed when you define the transaction to APPC.

Note that changing the transaction name in APPC implies that the new name must be assigned to `_FEKFSCMD_TP_NAME_` in `rsed.envvars`, as described in “`rsed.envvars`, RSE configuration file” on page 28.

Multiple LUs

APPC is a communication protocol that lets a program (the partner node) interact with a program on the host (the local node). With Developer for System z, both the partner node (TSO Commands server) and the local node (RSE server) are active on the same z/OS system. And by default, they both use the same (BASE) LU definition to communicate with each other.

You can specify an alternative partner LU name for the TSO Commands service in the `_FEKFSCMD_PARTNER_LU_` directive of `rsed.envvars`, as described in “`rsed.envvars`, RSE configuration file” on page 28. Note that you cannot change the local LU, which must always be a valid BASE LU (have the BASE and SCHED keywords).

LU security

VTAM supports a secure APPC setup, where the communication between the partner and local LU must be defined to the security software.

This is activated by adding `VERIFY=REQUIRED` to the VTAM definition of the local (BASE) LU. The security definitions must be done in the APPCLU class, as described in *MVS Planning: APPC/MVS Management* (SA22-7599).

Note that when this setup is active in VTAM, and the setup in your security software is not completed, the communication with the TSO commands service will fail to initialize without any message in the system log indicating that VTAM refused to set up the connection. The APPC IVP test (`fekfivpa`) will fail with message “Return code 1 - Allocate Failure no retry”.

Appendix E. Requisites

This appendix lists the host prerequisites and corequisites for this version of Developer for System z.

Refer to *Rational Developer for System z Prerequisites* (SC23-7659) in the Developer for System z online library at <http://www-01.ibm.com/software/awdtools/rdz/library/> for an up-to-date list of required and optional requisites.

The products listed in this section are all available at the time of publication for this manual. See the IBM Software Support Lifecycle Web site <http://www.ibm.com/software/support/lifecycle/>, to see whether a selected product is still available at the time that you want to use the related Developer for System z function.

z/OS host prerequisites

Use of Developer for System z requires that you have the following environment with the appropriate prerequisites:

z/OS

One of the following levels must be installed on the host:

Program Number	Product Name	PTFs or Service Levels Required
5694-A01	z/OS v 1.11	ISPF: <ul style="list-style-type: none">• APAR OA29489 (TSO/ISPF Client Gateway) PTF UA51713 TCP/IP: <ul style="list-style-type: none">• No PTF or Service Level required
5694-A01	z/OS v 1.10	ISPF: <ul style="list-style-type: none">• APAR OA29489 (TSO/ISPF Client Gateway) PTF UA51712 TCP/IP: <ul style="list-style-type: none">• APAR PK74282 (CSM fixed storage growth) PTF UK41810
5694-A01	z/OS v 1.9	ISPF: <ul style="list-style-type: none">• APAR OA29489 (TSO/ISPF Client Gateway) PTF UA51687 TCP/IP: <ul style="list-style-type: none">• APAR PK74282 (CSM fixed storage growth) PTF UK41812

Program Number	Product Name	PTFs or Service Levels Required
5694-A01	z/OS v 1.8	ISPF: <ul style="list-style-type: none"> • APAR OA20345 (nested command output) PTF UA33575 • APAR OA20449 (add NESTMACS support) PTF UA34052 • APAR OA29489 (TSO/ISPF Client Gateway) PTF UA51686 TCP/IP: <ul style="list-style-type: none"> • APAR PK74282 (CSM fixed storage growth) PTF UK41811

The related product Web site is:

<http://www-03.ibm.com/systems/z/os/zos/>

Notes:

1. Remote (host-based) actions for z/OS UNIX subprojects require that the z/OS UNIX version of REXEC or SSH is active on the host.
2. z/OS includes the following components, which need to be installed, configured, and operational:
 - Interactive System Productivity Facility (ISPF)
 - <http://www-01.ibm.com/software/awdtools/ispf/>
 - Language Environment
 - <http://www-03.ibm.com/servers/eserver/zseries/zos/le/>
 - RACF or equivalent security product
 - <http://www-03.ibm.com/servers/eserver/zseries/zos/racf/>
 - VTAM component of IBM Communications Server
 - <http://www-01.ibm.com/software/network/commserver/zos/>
 - IP Services component of IBM Communications Server
 - <http://www-01.ibm.com/software/network/commserver/zos/>
 - Binder
 - APPC (optional)

Note:

- APPC is a mandatory prerequisite if the service for ISPF APAR OA29489 is not available on your host system.
- APPC can be replaced by the ISPF Client Gateway functionality delivered in ISPF for z/OS 1.10, and available in ISPF for z/OS 1.8 and 1.9 with the appropriate PTFs applied.

SMP/E

In order to install Developer for System z, one of the following levels must be installed:

Program Number	Product Name	PTFs or Service Levels Required
5655-G44	IBM System Modification Program Extended (SMP/E) for z/OS v 3.5	No PTF or Service Level required
5655-G44	IBM System Modification Program Extended (SMP/E) for z/OS v 3.4	No PTF or Service Level required

The related product Web site is:

<http://www-03.ibm.com/systems/z/os/zos/smpe/>

SDK for z/OS Java 2 Technology Edition

In order to support applications that use Remote Systems Explorer (RSE), one of the following levels must be installed on the host:

Program Number	Product Name	PTFs or Service Levels Required
5655-R32	IBM 64-bit SDK for z/OS, Java 2 Technology Edition, v 6.0	service release 7
5655-R31	IBM 31-bit SDK for z/OS, Java 2 Technology Edition, v 6.0	service release 7
5655-N99	IBM 64-bit SDK for z/OS, Java 2 Technology Edition, v 5.0	service release 11
5655-N98	IBM 31-bit SDK for z/OS, Java 2 Technology Edition, v 5.0	service release 11

The related product Web site is:

<http://www.ibm.com/servers/eserver/zseries/software/java/>

Note: The PTF for Developer for System z APAR PM07305 must be applied when using a 64-bit version of Java. The PTF is available via the Developer for System z recommend service page, <http://www-01.ibm.com/support/docview.wss?rs=2294&context=SS2QJ2&uid=swg27006335>.

z/OS host corequisites

The products listed in this section and other stated software are required to support specific features of Developer for System z. The Developer for System z workstation client can be successfully installed without these requisites. However, a stated host requisite must be installed and operational at runtime for the corresponding feature to work as designed.

z/OS

Program Number	Product Name	PTFs or Service Levels Required
5694-A01	z/OS v 1.11	HLASM No PTF or Service Level required XL C/C++ No PTF or Service Level required SCLM No PTF or Service Level required LE (PL/I) No PTF or Service Level required TN3270 No PTF or Service Level required
5694-A01	z/OS v 1.10	HLASM No PTF or Service Level required XL C/C++ No PTF or Service Level required SCLM No PTF or Service Level required LE (PL/I) No PTF or Service Level required TN3270 No PTF or Service Level required
5694-A01	z/OS v 1.9	HLASM No PTF or Service Level required XL C/C++ No PTF or Service Level required SCLM • APAR OA27379 (SCLM Search) PTF UA46330 + UA46331, UA46332, UA46333, UA46334 (language-dependent) LE (PL/I) No PTF or Service Level required TN3270 No PTF or Service Level required

Program Number	Product Name	PTFs or Service Levels Required
5694-A01	z/OS v 1.8	HLASM No PTF or Service Level required XL C/C++ No PTF or Service Level required SCLM <ul style="list-style-type: none"> • APAR OA21104 (informational build mode) PTF UA35046 + UA35056, UA35057, UA35058, or UA35059 (language-dependent) • APAR OA16924 (enhance SCLMINFO) PTF UA29772 + UA29922, UA29923, UA29924, or UA29925 (language-dependent) • APAR OA16804 (add surrogate userid support) PTF UA33524 + UA33533, UA33534, UA33535, or UA33536 (language-dependent) LE (PL/I) <ul style="list-style-type: none"> • APAR PK41552 (new PL/I messages for Developer for System z) PTF UK24482 (English) or UK24483 (Japanese) TN3270 No PTF or Service Level required

The related product Web site is:

<http://www-03.ibm.com/systems/z/os/zos/>

Note:

1. JES3 v 1.10 or higher is a corequisite for JES3 users who want to use the Job Monitor support for viewing the output of active jobs.
2. High Level Assembler (HLASM) must be installed on the host with the listed service level, in order to compile assembler programs developed or edited within Developer for System z.

The related product Web site is:

<http://www.ibm.com/software/awdtools/hlasm/>

3. The XL C/C++ compiler must be installed on the host with the listed service level, in order to compile C/C++ programs developed or edited within Developer for System z.

The related product Web site is:

<http://www.ibm.com/software/awdtools/czos/>

4. SCLM must be installed on the host with the listed service level, in order to support SCLM Developer Toolkit.

The related product Web site is:

<http://www.ibm.com/software/awdtools/sclmsuite/sclm/>

Note:

- APAR OA16804 is only required if you want to use secure build, promote, and deploy.
 - APAR OA26997 is only required for member security support.
 - APAR OA27379 is only required for member security support or SCLM search functionality.
5. Language Environment must be installed on the host with the listed service level, in order to support Enterprise Service Tools for PL/I.
- The related product Web site is:

<http://www-03.ibm.com/servers/eserver/zseries/zos/le/>

6. TN3270 must be installed on the host with the listed service level in order to support the Host Connect emulator. TN3270 is a part of the IP Services component of IBM Communications Server.
- The related product Web site is:

<http://www-01.ibm.com/software/network/commserver/zos/>

COBOL compiler

To compile COBOL programs developed or edited within Developer for System z, one of the following levels must be installed on the host:

Program Number	Product Name	PTFs or Service Levels Required
5655-S71	IBM Enterprise COBOL for z/OS v 4.2	No PTF or Service Level required
5655-S71	IBM Enterprise COBOL for z/OS v 4.1	No PTF or Service Level required
5535-G53	IBM Enterprise COBOL for z/OS v 3.4	No PTF or Service Level required

The related product Web site is:

<http://www.ibm.com/software/awdtools/cobol/zos/>

Note: IBM Enterprise COBOL for z/OS v 4.1 is required for Enterprise Service Tools to generate Compiled XML Conversion that uses the XMLSS-based XML PARSE capability in COBOL v 4.1.

PL/I compiler

To compile PL/I programs developed or edited within Developer for System z, one of the following levels must be installed on the host:

Program Number	Product Name	PTFs or Service Levels Required
5655-H31	IBM Enterprise PL/I for z/OS v 3.9	No PTF or Service Level required
5655-H31	IBM Enterprise PL/I for z/OS v 3.8	No PTF or Service Level required
5655-H31	IBM Enterprise PL/I for z/OS v 3.7	No PTF or Service Level required

Program Number	Product Name	PTFs or Service Levels Required
5655-H31	IBM Enterprise PL/I for z/OS v 3.6	No PTF or Service Level required

The related product Web site is:

<http://www.ibm.com/software/awdtools/pli/plizos/>

Debug Tool for z/OS

To support remote debugging from Developer for System z, one of the following levels must be installed on the host:

Program Number	Product Name	Programming Language	APARs, PTFs, or Service Levels Required
5655-V50	IBM Debug Tool for z/OS V10.1	COBOL, PL/I, C/C++, assembler, and additional features	all available maintenance
5655-U27	IBM Debug Tool for z/OS V9.1	COBOL, PL/I, C/C++, assembler, and additional features	all available maintenance
5655-S16	IBM Debug Tool Utilities and Advanced Functions for z/OS V8.1.0	COBOL, PL/I, C/C++, assembler, and additional features	all available maintenance
5655-S17	IBM Debug Tool for z/OS V8.1.0	COBOL, PL/I, Assembler, C/C++	all available maintenance

Note: Support for CICS debug configurations in IBM Rational Developer for System z v7.6.1 or higher requires IBM Debug Tool v10.1 or v9.1 (PTF number - UK52904).

The related product Web site is:

<http://www.ibm.com/software/awdtools/debugtool/>

Note: Debug Tool Utilities and Advanced Functions (DTU&AF) is a superset of Debug Tool.

Starting with version 9, Debug Tool for z/OS and Debug Tool Utilities and Advanced Functions have been merged into a single offering.

CICS Transaction Server

To support applications with embedded CICS statements, one of the following levels must be installed:

Program Number	Product Name	PTFs or Service Levels Required
5655-S97	IBM CICS Transaction Server for z/OS v 4.1	No PTF or Service Level required
5697-E93	IBM CICS Transaction Server for z/OS v 3.2	UK34221

Program Number	Product Name	PTFs or Service Levels Required
5697-E93	IBM CICS Transaction Server for z/OS v 3.1	UK15767, UK15764, UK11782, UK11294, UK12233, UK12521, UK15261, UK15271, UK34221, UK34078

The related product Web site is:

<http://www.ibm.com/software/http/cics/tserver/>

Note:

- The CICS Transaction Server requires additional configuration to work with the debug tool.
- The RESTful interface available in CICS Transaction Server v 4.1 or higher is required to support Application Deployment Manager, Service Component Architecture, and Enterprise Service Tools features that are new to IBM Rational Developer for System z v 7.6 or higher.
- CICS Transaction Server v 3.2 or higher is required to support many features of Enterprise Service Tools.
For the complete list of specifics on runtime requirements refer to the Enterprise Service Tools documentation the IBM Rational Developer for System z Information Center at <http://publib.boulder.ibm.com/infocenter/ratdevz/v7r6/>.
- CICS Transaction Server v 3.1 with service UK34221 is the minimum for Application Deployment Manager.

IMS

To support applications using IMS database and data communications, one of the following levels must be installed on the host:

Program Number	Product Name	PTFs or Service Levels Required
5635-A02	IBM IMS v 11.1	No PTF or Service Level required
5635-A01	IBM IMS v 10.1	No PTF or Service Level required
5655-J38	IBM IMS v 9.1	No PTF or Service Level required

The related product Web site is:

<http://www.ibm.com/software/data/ims/ims/>

Note:

- IMS requires additional configuration to work with the debug tool.
- Version 10.1 or higher of IMS, IMS Connect, and IMS SOAP Gateway are required for Enterprise Service Tools.

DB2 for z/OS

To support DB2, one of the following levels must be installed on the host:

Program Number	Product Name	PTFs or Service Levels Required
5635-DB2	IBM DB2 for z/OS v 9.1	No PTF or Service Level required

Program Number	Product Name	PTFs or Service Levels Required
5625-DB2	IBM DB2 Universal Database™ for z/OS v 8.1	No PTF or Service Level required

The related product Web site is:

<http://www.ibm.com/software/data/db2/zos/>

Rational Team Concert for System z

For Jazz-based source control using Developer for System z remote projects, the following level must be installed.

Program Number	Product Name	PTFs or Service Levels Required
5724-V82	Rational Team Concert for System z Server v 2.0	FMID HAHA200 – Team Server <ul style="list-style-type: none"> • UK54064 • UK54071 • UK54073 • UK54095 • UK54098 FMID HAHB200 – Toolkit <ul style="list-style-type: none"> • UK54065 • UK54066 • UK54099 FMID HAHC200 – Job Monitor <ul style="list-style-type: none"> • No PTF or Service Level required FMID HAHD200 – BuildForge Agent <ul style="list-style-type: none"> • UK54097

The related product Web site is:

<http://www-01.ibm.com/software/awdtools/rtc/>

Note: Rational Team Concert Server v 1.0 or Rational Team Concert for System z Server v 1.0.1 provides selective support for some Developer for System z functions such as local projects. We recommend Rational Team Concert for System z Server v 2.0.0.2 for a more integrated and full featured experience

File Manager

To support File Manager integration, one of the following levels must be installed on the host:

Program Number	Product Name	PTFs or Service Levels Required
5655-U29	IBM File Manager for z/OS v 10.1	<ul style="list-style-type: none"> • UK54428

The related product Web site is:

<http://www.ibm.com/software/awdtools/filemanager/>

Fault Analyzer

To support Fault Analyzer integration, the following levels must be installed on the host:

Program Number	Product Name	PTFs or Service Levels Required
5655-V51	IBM Fault Analyzer v 10.1	No PTF or Service Level required
5655-U28	IBM Fault Analyzer v 9.1	No PTF or Service Level required
5655-S15	IBM Fault Analyzer v 8.1	No PTF or Service Level required

The related product Web site is as follows:

<http://www.ibm.com/software/awdtools/faultanalyzer/>

REXX

To use SCLM Developer Toolkit, one of the following levels must be installed on the host:

Program Number	Product Name	PTFs or Service Levels Required
5695-014	IBM Library for REXX on zSeries v 1.4	No PTF or Service Level required
5695-014	IBM Library for REXX on zSeries Alternate Library v 1.4.0 (FMIDs HWJ9143, JWJ9144)	No PTF or Service Level required

A version of the REXX/370 Alternate Library is available from the product Web site:

<http://www.ibm.com/software/awdtools/rexx/rexxzseries/>

Ported tools

IBM Ported Tools for z/OS must be installed (in z/OS UNIX) to use sftp or scp to do secure deployment in SCLM Developer Toolkit.

A version of IBM Ported Tools for z/OS is available from the product Web site:

http://www-03.ibm.com/servers/eserver/zseries/zos/unix/port_tools.html

Ant

Apache Ant must be installed (in z/OS UNIX) to do JAVA/J2EE builds in SCLM Developer Toolkit.

Apache Ant is an open-source, Java-based build tool that you can download from the product Web site:

<http://ant.apache.org/>

Endevor®

CA Endevor® Software Change Manager Release 12 must be installed to use the Developer for System z Interface for CA Endevor® SCM.

CA Endevor® SCM is a product from CA. The related product web site is:

<http://www.ca.com/us/products/product.aspx?ID=259>

Bibliography

Referenced publications

The following publications are referenced in this document:

Table 52. Referenced publications

Publication title	Order number	Reference	Reference Web site
Java Diagnostic Guide	SC34-6650	Java 5.0	http://www.ibm.com/developerworks/java/jdk/diagnosis/
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	Developer for System z	http://www-306.ibm.com/software/awdtools/rdz/library/
Rational Developer for System z Common Access Repository Manager Developer's Guide	SC23-7660	Developer for System z	http://www-306.ibm.com/software/awdtools/rdz/library/
Rational Developer for System z Prerequisites	SC23-7659	Developer for System z	http://www-306.ibm.com/software/awdtools/rdz/library/
Rational Developer for System z Host Configuration Quick Start	GI11-9201	Developer for System z	http://www-306.ibm.com/software/awdtools/rdz/library/
Rational Developer for System z Host Planning Guide	GI11-8296	Developer for System z	http://www-306.ibm.com/software/awdtools/rdz/library/
SCLM Developer Toolkit Administrator's Guide	SC23-9801	Developer for System z	http://www-306.ibm.com/software/awdtools/rdz/library/
APPC and WebSphere Developer for System z	SC23-5885	Whitepaper	http://www-306.ibm.com/software/awdtools/rdz/library/
Communications Server IP Configuration Guide	SC31-8775	z/OS 1.9	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Communications Server IP Configuration Reference	SC31-8776	z/OS 1.9	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Communications Server IP Diagnosis Guide	GC31-8782	z/OS 1.9	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Communications Server IP System Administrator's Commands	SC31-8781	z/OS 1.9	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Communications Server SNA Network Implementation Guide	SC31-8777	z/OS 1.9	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Communications Server SNA Operations	SC31-8779	z/OS 1.9	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Cryptographic Services System SSL Programming	SC24-5901	z/OS 1.9	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/

Table 52. Referenced publications (continued)

Publication title	Order number	Reference	Reference Web site
DFSMS Macro Instructions for Data Sets	SC26-7408	z/OS 1.9	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
DFSMS Using data sets	SC26-7410	z/OS 1.9	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Language Environment Customization	SA22-7564	z/OS 1.9	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Language Environment Debugging Guide	GA22-7560	z/OS 1.9	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
MVS Initialization and Tuning Guide	SA22-7591	z/OS 1.9	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
MVS Initialization and Tuning Reference	SA22-7592	z/OS 1.9	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
MVS JCL Reference	SA22-7597	z/OS 1.9	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
MVS Planning APPC/MVS Management	SA22-7599	z/OS 1.9	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
MVS Planning Workload Management	SA22-7602	z/OS 1.9	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
MVS System Commands	SA22-7627	z/OS 1.9	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Security Server RACF Command Language Reference	SA22-7687	z/OS 1.9	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Security Server RACF Security Administrator's Guide	SA22-7683	z/OS 1.9	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
TSO/E Customization	SA22-7783	z/OS 1.9	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
TSO/E REXX Reference	SA22-7790	z/OS 1.9	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
UNIX System Services Command Reference	SA22-7802	z/OS 1.9	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
UNIX System Services Planning	GA22-7800	z/OS 1.9	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
UNIX System Services User's Guide	SA22-7801	z/OS 1.9	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Using REXX and z/OS UNIX System Services	SA22-7806	z/OS 1.9	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Resource Definition Guide	SC34-6430	CICSTS 3.1	http://www-03.ibm.com/systems/z/os/zos/bkserv/zapplsbooks.html
Resource Definition Guide	SC34-6815	CICSTS 3.2	http://www-03.ibm.com/systems/z/os/zos/bkserv/zapplsbooks.html
Resource Definition Guide	SC34-7000	CICSTS 4.1	https://publib.boulder.ibm.com/infocenter/cicsts/v4r1/index.jsp?topic=/com.ibm.cics.ts.home.doc/library/library_html.html
RACF Security Guide	SC34-6454	CICSTS 3.1	http://www-03.ibm.com/systems/z/os/zos/bkserv/zapplsbooks.html

Table 52. Referenced publications (continued)

Publication title	Order number	Reference	Reference Web site
RACF Security Guide	SC34-6835	CICSTS 3.2	http://www-03.ibm.com/systems/z/os/zos/bkserv/zapplsbooks.html
RACF Security Guide	SC34-7003	CICSTS 4.1	https://publib.boulder.ibm.com/infocenter/cicsts/v4r1/index.jsp?topic=/com.ibm.cics.ts.home.doc/library/library_html.html
Language Reference	SC27-1408	Enterprise COBOL for z/OS	http://www-03.ibm.com/systems/z/os/zos/bkserv/zapplsbooks.html

The following Web sites are referenced in this document:

Table 53. Referenced Web sites

Description	Reference Web site
Developer for System z Information Center	http://publib.boulder.ibm.com/infocenter/ratdevz/v7r6/index.jsp
Developer for System z Support	http://www-306.ibm.com/software/awdtools/rdz/support/
Developer for System z Library	http://www-306.ibm.com/software/awdtools/rdz/library/
Developer for System z home page	http://www-306.ibm.com/software/awdtools/rdz/
Developer for System z Recommended service	http://www-01.ibm.com/support/docview.wss?rs=2294&context=SS2QJ2&uid=swg27006335
Developer for System z enhancement request	https://www.ibm.com/developerworks/support/rational/rfe/
z/OS internet library	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
CICSTS Information Center	https://publib.boulder.ibm.com/infocenter/cicsts/v4r1/index.jsp
Download Apache Ant	http://ant.apache.org/
Java keytool documentation	http://java.sun.com/j2se/1.5.0/docs/tooldocs/solaris/keytool.html
CA support home page	https://support.ca.com/

Informational publications

The following publications can be helpful in understanding setup issues for requisite host components:

Table 54. Informational publications

Publication title	Order number	Reference	Reference Web site
ABCs of z/OS System Programming Volume 9 (z/OS UNIX)	SG24-6989	Redbook	http://www.redbooks.ibm.com/
System Programmer's Guide to: Workload Manager	SG24-6472	Redbook	http://www.redbooks.ibm.com/
TCPIP Implementation Volume 1: Base Functions, Connectivity, and Routing	SG24-7532	Redbook	http://www.redbooks.ibm.com/
TCPIP Implementation Volume 3: High Availability, Scalability, and Performance	SG24-7534	Redbook	http://www.redbooks.ibm.com/

Table 54. Informational publications (continued)

Publication title	Order number	Reference	Reference Web site
TCP/IP Implementation Volume 4: Security and Policy-Based Networking	SG24-7535	Redbook	http://www.redbooks.ibm.com/

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 runtime 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 multiuser 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

.

Documentation notices for IBM Rational Developer for System Z

© Copyright IBM Corporation - 2010

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

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:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
3-2-12, Roppongi, Minato-ku, Tokyo 106-8711 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:

Intellectual Property Dept. for Rational Software
IBM Corporation

3039 Cornwallis Road, PO Box 12195
Research Triangle Park, NC 27709
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 measurements 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 illustrate 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. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademark acknowledgments

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at www.ibm.com/legal/copytrade.shtml.

Rational are trademarks of International Business Machines Corporation and Rational Software Corporation, in the United States, other countries, or both.

Intel[®] and Pentium[®] are trademarks of Intel Corporation in the United States, or other countries, or both.

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

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

UNIX is a registered trademark of The Open Group in the United States and other countries.

Index

Special characters

- _RSE_CMDSERV_OPTS, Defining extra Java startup parameters with 41
- _RSE_JAVAOPTS, Defining extra Java startup parameters with 37
- _RSE_PORTRANGE 149
- /etc/inittab, z/OS UNIX initialization, INETD 295
- /etc/rc, z/OS UNIX INETD startup 295
- .dstoreMemLogging 128
- .dstoreTrace 128

A

- access method, Using the TSO/ISPF Client Gateway 243
- Access methods, TSO 243
- access requirements 8
- access to spool files, Conditional 156
- access to system libraries, Improve 225
- Actions against jobs - execution limitations 155
- Activating Common Access Repository Manager 45
- Activating the CA Endeavor® Software Configuration Manager Repository Access Manager 56
- Activating the PDS Repository Access Manager 55
- Activating the sample Repository Access Managers (RAMs) 55
- Activating the SCLM Repository Access Manager 55
- Activating the skeleton Repository Access Manager 56
- Address space count 197
- Address Space size 141
- address space size limit 205
- Adjust CRASERV.properties 53
- Adjust ISPF.conf 53
- ADM, customization 65
- admin, SCLMDT 79
- administrative utility for CICS administrators
 - functions provided 235
- administrative utility messages 240
- Administrative utility, migration notes 239
- ADNCSDAR, RESTful non-primary and web service non-primary 67
- ADNCSDRS, RESTful primary 67
- ADNCSDTX, RESTful change ID 68
- ADNCSDWS, web service primary 69
- ADNJSPAU, Administrative utility 235
- ADNJSPAU, CICS administrative utility 66
- ADNMSGHC, pipeline message handler 69
- ADNMSGHS, pipeline message handler 69

- ADNTXNC, RESTful change ID 68
- ADNVCRD, CRD repository 66
- ADNVMFST, manifest repository 70
- allocation exec (APPC transaction JCL), Using 247
- allocation exec, using 244
- alternate RSE connection method 94
- Alternative CARMA server startup 53
- alternative partner LU name, TSO Commands service 304
- alternative setup options, APPC and VTAM 303
- Ant
 - corequisites 314
- Ant, Install and customize 78
- Apache Ant
 - corequisites 314
- APF authorization
 - FEK.SFEKAUTH 166
- APF, authorization 139
- APPC access method, Using 246
- APPC alternative setup options 303
- APPC changes, Activating 303
- APPC transaction
 - troubleshooting APPC 143
- APPC transaction and TSO Commands service 142
- APPC transaction checklist 96
- APPC transaction for the TSO Commands service 95
- APPC transaction JCL, Basic customization 246
- APPC transaction JCL, Use existing ISPF profiles with 246
- APPC transaction JCL, Using an allocation exec 247
- APPC transaction logging 132
- APPC transaction, Implementation 97
- APPC transaction, preparation 95
- APPC transactions, setup with multiple Developer for System z 247
- APPC usage considerations 98
- APPC, Setting up 299
- APPCPMxx 301
- Application Deployment Manager (ADM) 177
- Application Deployment Manager security 233
- Application Deployment Manager, CICS Resource Definition editor 231
- Application Deployment Manager, CICS Resource Definition server 231
- Application Deployment Manager, customization 65
- Application Deployment Manager, customizing 231
- Application development 226
- application protection for RSE, Define 172
- applications supported 3
- ASCHPMxx 302

- ASCHPMxx (*continued*)
 - MAX 217
- ASSIZEMAX 169
- audit control
 - _RSE_HOST_CODEPAGE 153
 - audit.* options 153
 - daemon.log 153
 - enable.audit.log 153
- audit data
 - actions logged 153
- audit logging, managed by RSE daemon 152
- audit.log 128
- authentication by RSE daemon 160
- authentication by security software 159
- Authentication methods 147
- authentication, JES Job Monitor 148
- authentication, Setting up SSL and X.509 269
- availability, Port 103

B

- backing up configured files
 - Version 7.0 255
 - Version 7.1 255
 - Version 7.5 255
- Base resolver configuration files 285
- basic customization 13
- batch submit, CA Endeavor® SCM RAM startup using 57
- batch submit, CARMA server startup using 49
- Bidirectional language support 84
- blank spaces, syntax diagram 124
- BPXBATCH, INETD startup 295
- BPXPRMxx 222
 - INADDRANYCOUNT 215
 - MAXASSIZE 141, 169, 214
 - MAXFILEPROC 214
 - MAXMMAPAREA 214
 - MAXPROCSYS 144, 213
 - MAXPROCUSER 144, 213
 - MAXSOCKETS 215
 - MAXTHREADS 213
 - MAXTHREADTASKS 213
 - MAXUIDS 144, 214
- BPXPRMxx, set limits in
 - MAXASSIZE 14
 - MAXPROCUSER 14
 - MAXTHREADS 14
 - MAXTHREADTASKS 14

C

- CA Endeavor® SCM RAM 56
- CA Endeavor® SCM RAM startup, batch submit 57
- CA Endeavor® SCM RAM startup, CRASTART 60

- CA Endeavor® SCM RAM, customizing 62
- Cache management utilities, Java Virtual Machines (JVMs) 229
- cache security, Java Virtual Machines (JVMs) 228
- cache size limits, Java Virtual Machines (JVMs) 228
- CARMA and TCP/IP ports 151
- CARMA logging
 - rsecomm.log 131
- CARMA server startup, Alternative 46, 50, 53
- CARMA startup, batch submit 49
- CARMA tracing 136
- CARMA, activating 45
- CARMA, FMID HCMA710 266
- CARMA, RSE interface to 47
- CEE.SCEELPA
 - SYS1.PARMLIB(LPALSTxx) 226
- Certificate Authority validation
 - gskkyman 158
 - SAF key ring 158
 - TRUST, HIGHTRUST 158
- Certificate Revocation List (CRL), querying
 - CRL environment variables 159
 - rsed.envvars 159
- certificate, X.509 148
- certificates, client authentication using X.509 158
- Changes between version 7.0 and version 7.1 266
- changes, DB2 82
- changes, PROCLIB 82
- changes, Workload Manager 81
- characters, Nonalphanumeric in syntax diagram 124
- checklist, APPC transaction 96
- checklist, client 12
- checklist, Client 11
- checklist, requirements 13
- checklist, SCLM administrator 79
- CICS administrative utility 66
- CICS Bidirectional language support 84
- CICS non-primary connection region 70
- CICS non-primary connection regions 67
- CICS primary connection region 67, 69
- CICS Resource Definition (CRD) editor, Application Deployment Manager 231
- CICS Resource Definition (CRD) server, Application Deployment Manager 231
- CICS resource definitions, administrator 231
- CICS resource definitions, developer 231
- CICS resource install logging 232
- CICS Transaction Server
 - corequisites 311
- CICS transactions 162
- CICSplex SM Business Application Services (BAS) 232
- CICSTS considerations 231
- CICSTS security 161
- class sharing between Java Virtual Machines (JVMs) 228
- class sharing, enabling in Java Virtual Machines (JVMs) 228
- classification rules, WLM 188
- CLASSPATH 250
- client authentication support, add X.509 277
- Client authentication using X.509 certificates 158
- Client checklist 11, 12
- Client Gateway access method, Using the TSO/ISPF 243
- Client Gateway configuration file, TSO/ISPF 42
- cloning existing RSE setup 272
- COBOL
 - remote check 136
- COBOL Compiler
 - corequisites 310
- coexistence, update rsed.envvars to enable coexistence 273
- command security, Define JES 170
- commands, JES Job Monitor Modify 117
- commands, JES Job Monitor Start 115
- commands, Lock daemon Modify 121
- commands, Lock daemon Start 116
- commands, Modify (F) 117
- commands, Operator 115
- commands, RSE daemon Modify 118
- commands, RSE daemon Start 115
- commands, Start (S) 115
- commands, Stop (P) 121
- COMMNDxx, add started tasks 15
- Common Access Repository Manager (CARMA), FMID HCMA710 266
- Common Access Repository Manager logging 131
- Common Access Repository Manager, Activating 45
- Communication encryption using SSL 149
- communication, External 150
- communication, Internal 151
- communication, SSL encrypted 157, 234
- component overview, Developer for System z
 - graphical representation 177
- components, Installing and configuring 3
- components, Optional 43
- Conditional access to spool files 156
- Conditional actions against jobs 153
- configurable files 260, 267
- configuration file changes, other products 255
- configuration files, Base resolver 285
- configuration files, Developer for System z 162
- configuration files, Identical software level, different 250
- configuration information, search orders of 284
- configuration of requisite products and software 8
- configuration problems, Troubleshooting 127
- configuring host components 3
- Connection flow 182
- Connection flow (*continued*)
 - graphical representation 182
- Connection refused 144
- connection regions, CICS
 - non-primary 67
- connection regions, primary versus non-primary 232
- Connection security 148
- connection, JES Job Monitor 105
- connection, lock daemon 105
- connection, REXEC 109
- connection, RSE daemon 104
- considerations, Performance 225
- considerations, pre-configuration 8
- considerations, predeployment 11
- considerations, Preinstallation 4
- considerations, Security 147
- considerations, server 9
- considerations, User ID 8
- console messages 121
 - JES Job Monitor 121
- Console messages
 - lock daemon 122
 - RSE daemon 122
 - RSE thread pool server 122
- controlled libraries for RSE, Define MVS 172
- corequisites
 - Ant 314
 - Apache Ant 314
 - CICS Transaction Server 311
 - COBOL Compiler 310
 - DB2 312
 - Debug Tool 311
 - Fault Analyzer 314
 - File Manager 313
 - host 307
 - IMS 312
 - Java 2 Technology Edition 307
 - PL/I Compiler 310
 - Ported tools 314
 - Rational Team Concert for System z 313
 - REXX 314
 - SDK for z/OS 307
 - z/OS 308
- corequisites, Developer for System z 305
- CRA\$VDEF, CARMA components 46
- CRA\$VMSG, CARMA components 46, 57
- CRA\$VSTR, CARMA components 46
- CRA#ASLM, sample SCLM RAM 55
- CRA#CIRX, IRXJCL versus CRAXJCL 63, 64
- CRA#CRAM, sample skeleton RAM 56
- CRA#UADD, CARMA components 46, 57
- CRA#UQRY, CARMA components 46, 57
- CRA#VCAD, Endeavor SCM RAM 57
- CRA#VCAS, Endeavor SCM RAM 57
- CRA#VPDS, sample PDS RAM 55
- CRA#VSLM, sample SCLM RAM 55
- CRAISPRX, ISPF alternative CARMA startup 53
- CRAISPRX, ISPF configuration file 42
- CRANDVRA, customizing 61

- CRASERV.properties, Adjust 53
- CRASRV.properties
 - clist.dsname 47
 - crastart.configuration.file 47
 - crastart.steplib 47
 - crastart.stub 47
 - crastart.syslog 47
 - crastart.tasklib 47
 - crastart.timeout 47
 - port.range 47
 - port.start 47
 - startup.script.name 47
- CRASRV.properties, adjust 49, 51
- CRASRV.properties, adjusting 57, 60
- CRASTART, alternative CARMA server
 - startup using 50
- CRASTART, CA Endeavor® SCM RAM
 - startup using 60
- crastart.conf, adjust 51
- crastart.conf, CARMA crastart
 - startup 51
- crastart.endevor.conf, adjusting 60
- CRASUBCA, adjusting 58
- CRASUBMT, CARMA batch startup 49
- CRAXJCL, IRXJCL versus CRAXJCL 63
- CRD repository 66, 161
- CRD repository security 233
- CRD server transaction IDs,
 - customizing 68
- CRD server using the RESTful
 - interface 67
- creating LSTRANS.FILE 76
- cron, WORKAREA directory cleanup 98
- customization - ISPF.conf, 243
- customization setup 13
- customization tasks, optional 81
- customization, basic 13
- customization, SCLM Developer
 - Toolkit 73
- customizing Application Deployment
 - Manager 231
- customizing CRD server transaction
 - IDs 68
- Customizing the TSO environment 243

D

- daemon, Lock 20
- daemon, RSE 19
- data set profiles, Define 166
- data sets fails, Opening MVS 144
- DB2
 - corequisites 312
- DB2 changes 82
- DB2 stored procedure 81
- Debug Tool
 - corequisites 311
- Debug Tool, IBM, corequisite 18
- Define MVS program controlled libraries
 - for RSE 172
- Define PassTicket support for RSE 173
- Define Port Of Entry checking for
 - RSE 161
- Define RSE server as a secure z/OS
 - UNIX 171
- Define z/OS UNIX program controlled
 - files for RSE 174

- definitions available to resolver 288
- definitions, prerequisite LINKLIST and
 - LPA 17
- definitions, Security 23, 163
- dependency, Hostname 283
- Developer for System z started tasks,
 - Define 169
- Developer for System z, component
 - overview
 - graphical representation 177
- Developer for System z,
 - understanding 177
- development, Application 226
- Diagnostic IRZ error messages 84
- different configuration files with identical
 - software levels 250
- directory cleanup, WORKAREA
 - skulker 98
- directory structure, z/OS UNIX
 - graphical representation 185
- Disk space, Java Virtual Machines
 - (JVMs) 229
- Dump files 133
- dump locations, z/OS UNIX 134
- dumps, Java 133
- dumps, MVS 133

E

- ELAXF* high level qualifier checklist 23
- ELAXF* procedures, Sample
 - ELAXFADT 22
 - ELAXFASM 22
 - ELAXFBMS 22
 - ELAXFCOC 22
 - ELAXFCOP 22
 - ELAXFCOT 22
 - ELAXFCP1 22
 - ELAXFCPC 22
 - ELAXFCPP 22
 - ELAXFDCL 22
 - ELAXFGO 22
 - ELAXFLNK 22
 - ELAXFMFS 22
 - ELAXFPL1 22
 - ELAXFPLP 22
 - ELAXFPLT 22
 - ELAXFPP1 22
 - ELAXFTSO 22
 - ELAXFUOP 22
- ELAXF* remote build procedures 22
- ELAXMJCL, DB2 changes 82
- ELAXMSAM 82
- Emulator, Host Connect 145
- enable class sharing, Java Virtual
 - Machines (JVMs) 228
- encrypted communication, SSL 157, 162,
 - 234
- encryption using SSL,
 - Communication 149
- encryption, ssl.properties 85
- Enterprise Service Tools support 83
- environmental settings, INETD 297
- Error feedback tracing 136
- error messages, IRZ Diagnostic 84
- EST support 83

- ETC.SERVICES
 - aliases 292
 - port_number 292
 - protocol 292
 - service_name 292
- execution limitations, Actions against
 - jobs 155
- existing ISPF profiles (APPC transaction
 - JCL), Use 246
- External communication 150
- external communication to specified
 - ports, limiting 149

F

- fa.log 128
- failure of MVS data sets to open 144
- Fault Analyzer
 - corequisites 314
- Fault Analyzer Integration logging
 - fa.log 131
 - rsecomm.log 131
- feedback tracing, Error 136
- FEJJCNFG 151, 222, 251
 - CONSOLE_NAME 155
 - MAX_THREADS 216
- FEJJCNFG, JES Job Monitor 162
- FEJJCNFG, JES Job Monitor configuration
 - file 24
- FEJJJCL, PROCLIB changes, JES Job
 - monitor 19
- FEJTSO 24
- FEKAPPC, APPC implementation 97
- FEKAPPCL, APPC implementation 97
- FEKAPPCX, APPC implementation 97
- FEKAPPL 148
- fekfivpi IVP test logging
 - fekfivpi.log 132
- fekfivpi.log 128
- fekfivpi.log, IVP test logging 132
- fekfivps.log, IVP test logging 132
- FEKFRSRV 97
- FEKLOCKD 20
- FEKLOGS, log and setup analysis
 - using 127
- FEKRACE, security definitions 23, 163
- fekrivp 139
- FEKRSED 20
- FEKSETUP 13, 67, 74, 97
- ffs.log 128
- ffsget.log 128
- ffsput.log 128
- File Manager
 - corequisites 313
- File Manager integration 91
- File Manager Integration logging
 - rsecomm.log 131
- file system attribute, SETUID 137
- file system space usage, z/OS UNIX 210
- file systems, zFS 225
- Fixed Java heap size 227
- FMID HCMA710 266
- FMID HHOP710 266
- FMID HHOP750 263
- FMID HHOP760 258
- FMIEXT.properties
 - enabled 92

FMIEXT.properties (*continued*)
 fmlistenport 92
 fragments, Syntax 125
 freeing a lock
 RSE , modify cancel command 184

G

goals, setting in WLM 189
 gskkyman, Create a key database
 with 277

H

Handler, Pipeline message 69
 HCMA710 266
 heap size limit, Java 205
 HHOP710 266
 HHOP750 263
 HHOP760 258
 host
 prerequisites 305
 host address not resolved, TCP/IP
 Resolver
 lock.log 289
 host based projects 90
 host based property groups 89
 host components, Installing and
 configuring 3
 Host Connect Emulator 145
 host corequisites 307
 host prerequisites 305
 host tables, Local 286
 Hostname dependency 283
 hostnames, applying to Developer for
 System z 286

I

IBM Debug Tool, corequisite 18
 ID considerations, User 8
 Identical setup across a sysplex 249
 identical software levels with different
 configuration files 250
 IEASYSxx 222
 MAXUSER 144, 216
 Improve access to system libraries 225
 Improving performance of security
 checking 226
 IMS
 corequisites 312
 INETD environmental settings 297
 INETD security 296
 INETD temporary file
 /etc/inetd.pid 294
 INETD, Developer for System z
 requirements 297
 INETD, setting up 291
 inetd.conf 291
 protocol 291
 server_program 291
 server_program_arguments 291
 service_name 291
 socket_type 291
 userid 291
 wait_flag 291

initialization, IVP 102
 install logging, CICS resource 232
 installation verification 99
 installation verification programs 101
 Installing and configuring host
 components 3
 integration, File Manager 91
 interface, RESTful versus Web
 Service 67
 Internal communication 151
 IRXJCL, IRXJCL versus CRAXJCL 63
 ISPSISPLD
 ISPF TSO/ISPF Client Gateway 172
 ISPF commands 42, 53
 ISPF profiles (APPC transaction
 JCL) 246
 ISPF profiles, Use existing 244
 ISPF TSO/ISPF Client Gateway
 ISPSISPLD 172
 ISPF, Use multiple allocation execs 245
 ISPF.conf, Adjust 53
 ISPF.conf 42
 allocjob 42
 ISPF_timeout 42
 isplib 42
 ispmlib 42
 ispplib 42
 ispslib 42
 isptlib 42
 sysproc 42
 ISPF.conf files, use with multiple
 setups 245
 ISPF.conf, Basic customization 243
 ISPF.conf, updates for SCLMDT 74
 ISPF's TSO/ISPF Client Gateway
 connection, verifying 106
 IVP
 fekfivpa 107
 fekfivpd 104
 fekfivpi 106
 fekfivpj 105
 fekfivpl 105
 fekfivpr 109
 fekfivps 108
 fekfivpz 110
 IVP initialization
 ivpinit 102
 IVP test logging
 fekfivpi.log 132
 fekfivps.log 132
 IVP, basic and optional services
 fekfivpa 101
 fekfivpd 101
 fekfivpi 101
 fekfivpj 101
 fekfivpl 101
 fekfivpr 101
 fekfivps 101
 fekfivpt 101
 fekfivpz 101
 IVP scripts 101
 ivpinit shell script, setting RSE
 environment variables with 102
 IVTPRMxx
 ECSA MAX 216
 FIXED MAX 216

J

J2EE 73, 78
 Java 73, 78
 Java 2 Technology Edition
 corequisites 307
 Java dumps 133
 Java heap size limit 205
 Java heap size, Fixed 227
 Java startup parameters with
 _RSE_CMDSERV_OPTS, Defining 41
 Java startup parameters with
 _RSE_JAVAOPTS, Defining 37
 Java Virtual Machines (JVMs), class
 sharing between 228
 Java Xquickstart option 227
 JAVA_DUMP_TDUMP_PATTERN 133
 JCL requirements, startup 141
 JES command security, Define 170
 JES JMON
 GEN_CONSOLE_NAME 156
 JES JMON, FEJJC�FG
 _BPXK_SETIBMOPT
 _TRANSPORT 25
 APPLID 25
 AUTHMETHOD 25
 CODEPAGE 25
 CONCHAR 25
 CONSOLE_NAME 25
 GEN_CONSOLE_NAME 25
 HOST_CODEPAGE 25
 LIMIT_COMMANDS 25
 LIMIT_VIEW 25
 LISTEN_QUEUE_LENGTH 25
 MAX_DATASETS 25
 MAX_THREADS 25
 SERV_PORT 25
 SUBMITMETHOD 25
 TIMEOUT 25
 TIMEOUT_INTERVAL 25
 TSO_TEMPLATE 25
 TZ 25
 JES Job Monitor (JMON) 177
 JES Job Monitor authentication 148
 JES Job Monitor configuration
 GEN_CONSOLE_NAME 156
 JES Job Monitor configuration file,
 FEJJC�FG, 24
 JES Job Monitor connection 105
 JES Job Monitor logging 129
 JES Job Monitor server 19
 JES Job Monitor started task, JMON 19
 JES Job Monitor tracing 135
 JES Job Monitor, FEJJC�FG 162
 JES Job Monitor, JMON 99
 JES Job Monitor, Modify command 117
 JES Job Monitor, Start command 115
 JES security 153
 JMON 19, 170, 251
 fekfivpj 105
 JMON, JES Job Monitor 99
 Job Monitor server, JES 19
 jobs, Conditional actions against 153
 JVMs, class sharing between 228

K

- key database, Create with gskkyman 277
- key resource definitions 212
 - rsed.envvars 212
 - SYS1.PARMLIB(BPXPRMxx) 213
- key ring, Create with RACF 271
- key store with keytool, Create 280
- keytool, Create a key store with 280

L

- Language Environment runtime libraries 225
- libraries for RSE , Define MVS 172
- libraries, Improve access to system 225
- libraries, Language Environment runtime 225
- LIMIT_COMMANDS 154
- LIMIT_VIEW 156
- limiting external communication, specified ports 149
- limits, System 144
- limits, z/OS UNIX set in BPXPRMxx 14
- LINKLIST definitions, prerequisite 17
- LINKLIST, definitions for other products 18
- Local definitions available to resolver 288
- Local host tables 286
- lock daemon
 - console messages 122
- Lock daemon 20, 183
- Lock Daemon (LOCKD) 177
- lock daemon connection 105
- Lock daemon flow
 - graphical representation 183
- Lock daemon logging 129
- lock daemon tracing 136
- lock daemon, LOCKD 99
- Lock daemon, Modify command 121
- Lock daemon, Start command 116
- lock.log 128
- LOCKD 9
- LOCKD, lock daemon 99
- Log and setup analysis using FEKLOGS 127
- log files
 - .dstoreMemLogging 128
 - .dstoreTrace 128
 - audit.log 128
 - fa.log 128
 - fekfivpi.log 128
 - fekfivps.log 128
 - ffs.log 128
 - ffsget.log 128
 - ffsput.log 128
 - lock.log 128
 - rmt_class_loader.cache.jar 128
 - rsecomm.log 128
 - rsedaemon.log 128
 - rseserver.log 128
 - serverlogs.count 128
 - stderr.log 128
 - stdout.log 128

- Logging configuration file, rsecomm.properties 88
- logging, APPC transaction 132
- logging, CARMA 131
- logging, Fault Analyzer Integration 131
- logging, fekfivpi IVP test 132
- logging, File Manager Integration 131
- logging, JES Job Monitor 129
- logging, Lock daemon 129
- logging, RSE daemon 129
- logging, RSE user 130
- logging, SCLM Developer Toolkit 131
- logging, thread pool 129
- long/short name translation, rsed.envvars updates for 78
- Long/short name translation, SCLM 75
- LPA definitions, prerequisite 17
- LPALSTxx 226
- LPALSTxx, LPA definitions in 15
- LSTRANS.FILE, creating 76
- LU security, VTAM 304

M

- management, Workload 227
- manifest repository 70
- Message handler, Pipeline 69
- messages, administrative utility 240
- messages, console 121
- methods, Authentication 147
- migration 255
- migration considerations 255
- Migration considerations 3
- migration notes, administrative utility 239
- migration, version 7.5 to 7.6 258
- Modify (F) command 117
- monitoring RSE 217
- monitoring z/OS UNIX 218
- monitoring z/OS UNIX file systems 220
- monitoring, network 220
- multiple allocation execs, TSO/ISPF 245
- Multiple APPC transactions 247
- multiple Developer for System z setups, use multiple APPC transactions with 247
- multiple Developer for System z setups, use multiple ISPF.conf files with 245
- multiple instances, Running 249
- Multiple ISPF.conf files 245
- multiple LUs 304
- MVS data sets fails on opening 144
- MVS dumps 133
- MVS program controlled libraries for RSE , Define 172

N

- name translation, SCLM 75
- netstat 140
- netstat, TCP/IP setup 103
- network, monitoring 220
- non-primary connection regions, CICS 67
- non-system administrators, update privileges 186

- Nonalphanumeric characters, syntax diagram 124

O

- OMVS segment, Define 166
- one-time password and User ID 148
- operand, Selecting more than one in a syntax diagram 124
- Operands, syntax diagram 124
- operating systems supported 3
- Operator commands 115
- Optional components 43
- optional customization tasks 81

P

- PARM variable, JCL limitations 21
- PARMLIB, changes 14
- PassTicket support for RSE , Define 173
- PassTickets, using 152
- password and User ID 148
- PDS Repository Access Manager, Activating 55
- Performance considerations 225
- performance of security checking, Improving 226
- permission bits, z/OS UNIX 137
- Pipeline message handler 69
- Pipeline security 233
- PL/I Compiler
 - corequisites 310
- platforms supported 3
- POE checking 149, 161
- Port availability 103
- port definitions, PROFILE.TCPIP 294
- Port of Entry checking 161
- Port Of Entry checking 149
- Ported tools
 - corequisites 314
- PORTRANGE 37, 140
- PORTRANGE available for RSE , Defining 36
- ports, CARMA and TCP/IP 151
- ports, limiting external communication to specific 149
- ports, Reserved TCP/IP 140
- ports, REXEC 94
- ports, TCP/IP 150
- pre-configuration considerations 8
- predeployment considerations 11
- Preinstallation considerations 4
- prerequisite products 5
- prerequisites
 - host 305
 - SMP/E 306
- prerequisites, Developer for System z 305
- previously configured files, backing up
 - Version 7.0 255
 - Version 7.1 255
 - Version 7.5 255
- primary connection region, CICS 67
- primary versus non-primary connection regions 232

- private keys and certificates, decide where to store 269
- Process count 200
- PROCLIB changes 82
- PROCLIB, changes 19
- products and software, configuration of requisite 8
- products, prerequisite 5
- PROFILE.TCPIP port definitions 294
- profiles, Define data set 166
- Program Control authorization 137
- PROGxx, APF authorizations 16
- PROGxx, LINKLIST definitions 16
- projectcfg.properties 90
 - PROJECT-HOME 91
 - WSED-VERSION 91
- projects, host based 90
- propertiescfg.properties 89
 - DEFAULT-VALUES 90
 - ENABLED 90
 - PROPERTY-GROUP 90
 - RDZ-VERSION 90
- property groups, host based 89
- publications, Referenced 317

Q

- qualifier checklist, ELAXF* 23
- querying a Certificate Revocation List (CRL)
 - CRL environment variables 159
 - rsed.envvars 159
- quickstart, Java option (-Xquickstart) 227

R

- RACF
 - permits 167
- RACF, Create a key ring with 271
- Rational Team Concert for System z
 - corequisites 313
- read a syntax diagram, How to 123
- Referenced publications 317
- refused connection 144
- remote build procedures, ELAXF* 22
- Remote Execution, using 93
- Remote host-based actions, z/OS UNIX subprojects 94
- Remote Systems Explorer 10
- Remove old files from WORKAREA 80
- Repository Access Manager 56
- Repository Access Manager, Activating PDS 55
- Repository Access Manager, Activating SCLM 55
- Repository Access Manager, Activating skeleton 56
- Repository Access Manager, Activating the CA Endevor® SCM 56
- Repository Access Managers (RAMs), Activating sample 55
- repository security, CRD 233
- repository, CRD 66
- repository, manifest 70
- required resources 5

- requirements checklist 13
- requirements, startup JCL 141
- requisite issues, known 144
- requisite products and software, configuration of 8
- Requisites, Developer for System z 305
- Reserved TCP/IP ports 140
- resolver, Local definitions available to 288
- resolvers, Understanding 284
- resource definitions, various 216
- resource install logging, CICS 232
- resource security 234
- resource usage, overview 196
- resource usage, tuning 195
- resources, required 5
- RESTful interface 232
 - ADMI 68
 - ADMR 68
 - ADMS 68
- RESTful interface versus Web Service interface 67, 232
- RESTful interface, CRD server using 67
- REXEC connection, verifying 109
- REXEC daemon, user ID permissions when started by INETD 298
- REXEC set up 94
- REXEC, using 93
- REXEC/SSH shell script 110
- REXX
 - corequisites 314
 - rmt_class_loader_cache.jar 128
- RSE 10
- RSE , Define MVS program controlled libraries for 172
- RSE , Define PassTicket support for 173
- RSE , Define Port Of Entry checking for 161
- RSE , Defining the PORTRANGE 36
- RSE as a Java application
 - graphical representation 179
- RSE configuration file, rsed.envvars, 28
- RSE connection method, alternate 94
- RSE daemon 9, 19, 150
 - console messages 122
- RSE daemon (RSED) 177
- RSE daemon and audit logging 152
- RSE daemon connection 104
- RSE daemon log files
 - audit.log 129
 - rsedaemon.log 129
 - rseserver.log 129
 - serverlogs.count 129
 - stderr.*.log 129
 - stdout.*.log 129
- RSE daemon logging 129
- RSE daemon, authentication by 160
- RSE daemon, Modify command 118
- RSE daemon, RSED 99
- RSE daemon, Start command 115
- RSE interface to CARMA 47
- RSE server 150
- RSE setup, Clone existing 272
- RSE SSL encryption, ssl.properties, daemon properties 85
 - server properties 85
- RSE started task, RSED 20

- RSE thread pool log files
 - audit.log 129
 - rsedaemon.log 129
 - rseserver.log 129
 - serverlogs.count 129
 - stderr.*.log 129
 - stdout.*.log 129
- RSE thread pool server
 - console messages 122
- RSE trace configuration, rsecomm.properties, 88
- RSE tracing 135
- RSE user logging
 - .dstoreMemLogging 130
 - .dstoreTrace 130
 - ffs.log 130
 - ffsget.log 130
 - ffsput.log 130
 - lock.log 130
 - rmt_class_loader.cache.jar 130
 - rsecomm.log 130
 - stderr.log 130
 - stdout.log 130
- RSE, define application protection for 172
- RSE, Define as a secure z/OS UNIX server 171
- RSE, Define z/OS UNIX program controlled files for 174
- RSE, monitoring 217
- RSE, rsed.envvars
 - _RSE_JAVAOPTS 162
- RSE, ssl.properties 163
- rsecomm.log 128
 - File Manager Integration logging 131
 - SCLM Developer Toolkit logging 131
- rsecomm.properties 135
 - debug_level 88
 - log_location 88
 - server.version 88
- rsecomm.properties, 88
- RSED 20
- RSED, RSE daemon 99
- rsed.envvars 118, 212, 250
 - _BPX_SHAREAS 35
 - _BPX_SPAWN_SCRIPT 35
 - _BPXK_SETIBMOPT
 - _TRANSPORT 34
 - _CEE_DMPTARG 32
 - _CEE_RUNOPTS 35
 - _CMDSERV_BASE_HOME 33
 - _CMDSERV_CONF_HOME 33, 246
 - _CMDSERV_WORK_HOME 33
 - _FEKFSCMD_PARTNER_LU_ 34
 - _FEKFSCMD_TP_NAME_ 34
 - _RSE_CMDSERV_OPTS 35
 - _RSE_DAEMON_CLASS 35
 - _RSE_HOST_CODEPAGE 32
 - _RSE_JAVAOPTS 35, 133, 243
 - _RSE_LOCKD_CLASS 35
 - _RSE_LOCKD_PORT 32
 - _RSE_POOL_SERVER_CLASS 35
 - _RSE_PORTRANGE 34, 149
 - _RSE_SERVER_CLASS 35
 - _RSE_SERVER_TIMEOUT 35
 - _SCLMDT_TRANSTABLE 34
 - &ISPROF=&SYSUID..ISPPROF= 42

rsed.envvars (continued)

ANT_HOME 34
 CGI_DTWORk 35
 CLASSPATH 35
 DAPPLID 41
 Daudit.cycle 39
 Daudit.retention.period 39
 Ddaemon.log 37
 DDENY_PASSWORD 41
 Ddeny.nonzero.port 40
 DDSTORE_IDLE_SHUTDOWN_TIMEOUT 41
 DDSTORE_LOG_DIRECTORY 37
 DDSTORE_MEMLOGGING_ON 41
 DDSTORE_TRACING_ON 41
 Denable.audit.log 39
 Denable.automount 39
 Denable.certificate.mapping 39
 Denable.port.of.entry 39
 Denable.standard.log 39
 DHide_ZOS_UNIX 41
 Dipv6 39
 Dkeep.last.log 39
 Dmaximum.clients 37, 212
 Dmaximum.threadpool.process 39, 212
 Dmaximum.threads 37, 212
 Dminimum.threadpool.process 37, 212
 Dprocess.cleanup.interval 40
 Dsingle.logon 40
 DSTORE_LOG_DIRECTORY 131, 135
 DTso_SERVER 41
 Duser.log 37
 GSK_CRL_SECURITY_LEVEL 34
 GSK_LDAP_PASSWORD 34
 GSK_LDAP_PORT 34
 GSK_LDAP_SERVER 34
 GSK_LDAP_USER 34
 JAVA_HOME 32
 JAVA_PROPAGATE 35
 LANG 32
 LIBPATH 35
 PATH 32, 35
 RSE_HOME 32
 RSE_JAVAOPTS 33
 RSE_LIB 35
 RSE_SAF_CLASS 33
 SCLMDT_BASE_HOME 35
 SCLMDT_CONF_HOME 34
 SCLMDT_WORK_HOME 35
 STEPLIB 32, 33, 34, 157
 TZ 32
 Xms 37, 212
 Xmx 37, 212
 rsed.envvars, RSE configuration file 28
 rsed.envvars, update to enable coexistence 273
 rsed.envvars, updates for long/short name translation 78
 rsed.envvars, updates for SCLMDT 75
 rsedaemon.log 128
 rserver.log 128
 Running multiple instances 249
 runtime libraries, Language Environment 225

S

sample setup 220
 defining limits 221
 determining minimum limits 221
 thread pool count 221
 sample storage, usage analysis 206
 SCLM 78
 SCLM administrator checklist 79
 SCLM Developer Toolkit 172
 SCLM Developer Toolkit (SCLMDT) 177
 SCLM Developer Toolkit connection, verifying
 SCLMDT checks 108
 SCLM Developer Toolkit logging
 rsecomm.log 131
 SCLM Developer Toolkit service 17
 SCLM Developer Toolkit, customization 73
 SCLM Repository Access Manager, Activating 55
 SCLM security 162
 SCLM updates for SCLMDT 79
 SCLM, Long/short name translation 75
 SCLMDT admin 79
 SCLMDT connection, verifying 108
 SCLMDT, ISPF.conf updates for 74
 SCLMDT, rsed.envvars updates for 75
 SDSF 74
 search order, native MVS environment 293
 search order, z/OS UNIX environment 293
 search orders of configuration information 284
 Search orders, z/OS UNIX environment 285
 secure APPC setup, VTAM 304
 Secure Shell, using 93
 Secure socket layer host configuration connection, Test 274
 Secure Socket Layer, Communication encryption using 149
 Secure Socket Layer, Setting up 269
 secure z/OS UNIX server, Define RSE as a 171
 security checking, Improving performance of 226
 Security considerations 147
 Security definitions 23, 163
 security definitions, Checklist 164
 security profile, Limitations stored in 142
 security settings and classes, Activate 165
 security settings, verify 174
 security software, authentication by security, Application Deployment Manager (ADM) 233
 security, CICSTS 161
 security, Connection 148
 security, Define JES command 170
 security, INETD 296
 security, JES 153
 security, pipeline 233
 security, resource 234
 security, SCLM 162
 security, transaction 233
 segment, Define OMVS 166
 server considerations 9
 serverlogs.count 128
 service connection, TSO Commands (APPC) 107
 setting goals, WLM 189
 settings and classes, Activate security 165
 SETUID file system attribute 137
 setup, customization 13
 setup, identical across a sysplex 249
 shell script, REXEC/SSH 110
 shell session, INETD startup 296
 signed certificate, self-signed or signed by Certificate Authority 271
 size estimate, guidelines 206
 size limit, address space 205
 size limit, Java heap 205
 size, Address Space 141
 skeleton Repository Access Manager, Activating 56
 skulker, WORKAREA directory cleanup 98
 SMP/E
 prerequisites 306
 SMP/E install, sticky bit 139
 Software Configuration Manager 56
 software level, identical in different configuration files 250
 software, configuration of requisite products and 8
 space usage, z/OS UNIX file system 210
 spaces, syntax diagram 124
 spool files, Conditional access to 156
 SSH daemon, user ID permissions when started by INETD 298
 SSH set up 94
 SSH shell script 110
 SSH, using 93
 SSL encrypted communication 157, 162, 234
 SSL host configuration connection, Test 274
 SSL, Communication encryption using 149
 SSL, Setting up 269
 ssl.properties 85
 daemon_key_label 86
 daemon_keydb_file 86
 daemon_keydb_password 86
 enable_ssl 86
 server_keystore_file 86
 server_keystore_label 86
 server_keystore_password 86
 server_keystore_type 86
 ssl.properties, activate SSL by creating new RSE daemon 273
 ssl.properties, Activate SSL by updating 273
 Start (S) command 115
 started tasks, Define for Developer for System z
 JMON started tasks 169
 LOCKD started tasks 169
 RSED started tasks 169
 started tasks, verifying 99
 started tasks, verifying services 101

- startup JCL requirements 141
- startup parameters with
 - _RSE_CMDSERV_OPTS, Defining extra Java 41
- startup parameters with
 - _RSE_JAVAOPTS, Defining extra Java 37
- startup, Alternative CARMA server 53
- startup, z/OS UNIX INETD 294
- stderr.*.log 128
- stderr.log 128
- stdout.*.log 128
- stdout.log 128
- STEPLIB use 18
- STEPLIB, Avoid use of 225
- sticky bit, MVS load module availability to z/OS UNIX 139
- Stop (P) command 121
- storage usage 205
- stored procedure, DB2 81
- subsystem types
 - ASCH 188
 - CICS 188
 - JES 188
 - OMVS 188
 - STC 188
- subsystems supported 3
- support for RSE, Define PassTicket 173
- support, Enterprise Service Tools 83
- support, EST 83
- Symbols, syntax diagram 123
- syntax diagram, How to read 123
- syntax diagram, Longer than one line 124
- syntax diagram, Nonalphanumeric characters and blank spaces 124
- syntax diagram, Operands 124
- syntax diagram, Selecting more than one operand 124
- syntax diagram, symbols 123
- Syntax fragments 125
- Syntax, example 124
- SYS1.PARMLIB(APPCPMxx) 301
- SYS1.PARMLIB(ASCHPMxx) 302
- SYS1.PARMLIB(BPXPRMxx) 222
 - MAXASSIZE 141, 169
 - MAXPROCSYS 144
 - MAXPROCUSER 144
 - MAXUIDS 144
- SYS1.PARMLIB(BPXPRMxx), Java Virtual Machines (JVMs) 229
- SYS1.PARMLIB(BPXPRMxx), Limitations set in 141
- SYS1.PARMLIB(IEASYSxx) 222
 - MAXUSER 144
- SYSEXEC 42
- sysplex, identical setup across 249
- SYSPROC 42
- system exits, Limitations enforced by 142
- system libraries, Improve access to 225
- System limits 144

T

- tables, Local host 286
- tables, Translate 285

- task owners 180
- tasks, optional customization 81
- TCP/IP ports 150
- TCP/IP ports, graphical representation 150
- TCP/IP ports, Reserved 140
- TCP/IP Resolver, host address not resolved
 - lock.log 289
- TCP/IP setup, netstat 103
- TCP/IP, applying to Developer for System z 286
- TCP/IP, Local definitions available to resolver 288
- TCP/IP, Setting up 283
- test logging, fekfivpi IVP 132
- Test the SSL host configuration connection 274
- third party and X.509 certificate 148
- Thread count 202
- thread pool logging 129
- thread security in RSE server
 - PassTickets 152
- TN3270 310
- trace configuration,
 - rsecomm.properties 88
- tracing 135
- tracing, CARMA 136
- tracing, Error feedback 136
- tracing, JES Job Monitor 135
- tracing, lock daemon 136
- tracing, RSE 88, 135
- transaction checklist, APPC 96
- transaction dump pattern variables 133
- transaction name, alternative to FEKFRSRV for TSO Commands service 304
- transaction security 233
- Translate tables 285
- Troubleshooting configuration problems 127
- TSO Access methods 243
- TSO Command Service 177
- TSO commands 53
- TSO Commands service 142, 243
- TSO Commands service connection (APPC) 107
- TSO Commands service logging 132
- TSO Commands service transaction, Defining 303
- TSO Commands service, APPC transaction for 95
- TSO environment, Customizing 243
- TSO/ISPF Client Gateway access method, Using 243
- TSO/ISPF Client Gateway configuration file 42
- TSO/ISPF, customization - ISPF.conf, 243
- TSO/ISPF, Use existing ISPF profiles 244
- TSO/ISPF, Use multiple allocation execs 245
- TSO/ISPF, use with multiple setups 245
- TSO/ISPF, Using an allocation exec 244
- tuning considerations 195

U

- uchars.settings, Uneditable characters 92
- understanding Developer for System z 177
- Uneditable characters, customizing to handle 92
- Uneditable characters, uchars.settings 92
- UNIX dump locations 134
- UNIX environment, Search orders used in 285
- UNIX program controlled files for RSE, Define 174
- UNIX server, Define RSE as 171
- update privileges, non-system administrators 186
- usage analysis, sample storage 206
- usage considerations, APPC 98
- use existing ISPF profiles 244
- use of STEPLIB, Avoid 225
- User ID and one-time password 148
- User ID and password 148
- User ID considerations 8
- user logging, RSE 130
- Using an allocation exec 244
- using batch submit for CARMA server startup 49
- using PassTickets 152

V

- Various resource definitions 216
 - EXEC card, server JCL 216
 - FEJJCNFG 216
 - SYS1.PARMLIB(ASCHPMxx) 217
 - SYS1.PARMLIB(IEASYSxx) 216
 - SYS1.PARMLIB(IVTPRMxx) 216
- Verify security settings 174
- verifying ISPF's TSO/ISPF Client Gateway connection 106
- verifying REXEC/SSH shell script 110
- version 7.0 and version 7.1, Changes between 266
- Virtual Telecommunications Access Method 300
- VSAM 299
- VTAM 300
- VTAM alternative setup options 303

W

- Web Owning Region 232
- Web Service interface 232
 - ADMI 69
 - ADMR 69
 - ADMS 69
- Web Service interface versus RESTful interface 67
- Web Service interface, CRD server using 68
- where to store private keys and certificates 269
- WLM classification rules 188
- WLM considerations xvi, 187
- WORKAREA directory cleanup skulker 98
- WORKAREA, Remove old files from 80

- workload classification, WLM 187
- Workload management 227
- workload manager 187
- Workload Manager changes 81

X

- x.509 authentication, setting up 269
- X.509 certificate 148
- X.509 certificates, client authentication
 - using 158
- X.509, adding client authentication support 277
- Xquickstart, Java option 227

Z

- z/OS
 - corequisites 308
- z/OS corequisites
 - C/C++ 309
 - High Level Assembler 309
 - Language Environment 310
 - SCLM 309
- z/OS host prerequisites 305
- z/OS projects perspective 90
- z/OS UNIX directory structure
 - graphical representation 185
- z/OS UNIX dump locations 134
- z/OS UNIX environment, Search orders
 - used in 285
- z/OS UNIX file system space usage 210
- z/OS UNIX file systems,
 - monitoring 220
- z/OS UNIX limits set in BPXPRMxx 14
- z/OS UNIX permission bits 137
- z/OS UNIX program controlled files for RSE, Define 174
- z/OS UNIX server, Define RSE as 171
- z/OS UNIX subprojects, Remote
 - host-based actions for 94
- z/OS UNIX, monitoring 218
- z/OShost
 - prerequisites 305
- zFS file systems, Using 225

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

IBM Rational Developer for System z
Host Configuration Guide
Version 7.6.1

Publication No. SC23-7658-04

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)

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



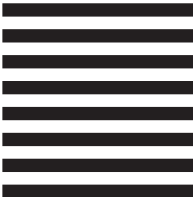
NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

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

SC23-7658-04

