

VisualAge Generator



Generation Guide

Version 4.5

Note

Before using this document, read the general information under “Notices” on page ix.

Fourth Edition (April 2001)

This edition applies to the following licensed programs:

- IBM VisualAge Generator Developer for OS/2 and Windows NT Version 4.5
- IBM VisualAge Generator Server for OS/2, AIX, Windows NT, HP-UX, and Solaris Version 4.5
- IBM VisualAge Generator Server for AS/400 Version 4 Release 4
- IBM VisualAge Generator Server for MVS, VSE, and VM Version 1.2

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 in any one of the following methods:

Electronically, using the online reader comment form at the address listed below. Be sure to include your entire network address if you wish a reply.

- <http://www.ibm.com/software/ad/visgen>

By mail to the following address:

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

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.

© **Copyright International Business Machines Corporation 1980, 2001. All rights reserved.**

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

Contents

Notices	ix
Trademarks	xi
Terminology used in this document	xiii
Terminology differences between Java and Smalltalk	xiv
About this document	xv
How to use this document	xv
Documentation provided with VisualAge Generator	xv

Part 1. Introducing generation 1

Chapter 1. Introducing program generation	3
Generating C++, Java, and COBOL programs	5
Preparing C++, Java, and COBOL programs	6

Part 2. Generating C++ programs. 9

Chapter 2. Inputs to C++ program generation	11
Inputs for C++ generation	11
Generation options parts	12
Linkage table parts.	12
Resource associations part	13
Conversion tables	13

Chapter 3. Outputs of C++ program generation	15
---------------------------------------------------------------	-----------

Chapter 4. Preparation process for C++ generation	17
Preparing C++ programs.	17
Analyzing preparation errors	18
Common preparation errors.	18

Chapter 5. Command interface for C++ generation	21
GENERATE subcommand syntax for C++ generation	21

GENERATE subcommand examples for C++.	24
START subcommand syntax.	26
STOP subcommand syntax	27
VALIDATE subcommand syntax for C++ generation	27

Part 3. Generating Java programs 29

Chapter 6. Inputs to Java server program generation	31
Inputs for Java generation	31
Generation Options parts	32
Linkage table parts.	32
Resource associations part	32
Conversion tables	33

Chapter 7. Outputs of Java program generation	35
----------------------------------------------------------------	-----------

Chapter 8. Preparation process for Java generation	39
Preparing Java programs.	39
Analyzing preparation errors	40
Common preparation errors.	40

Chapter 9. Command interface for Java generation	43
GENERATE subcommand syntax for Java generation	43
GENERATE subcommand examples for Java	45
START subcommand syntax.	46
STOP subcommand syntax	46
VALIDATE subcommand syntax for Java generation	47

Part 4. Generating COBOL programs 49

Chapter 10. Inputs to COBOL generation	51
Inputs for COBOL generation	51
Generation options parts.	53

Linkage table parts	53
Link edit parts	53
Reserved-word file	53
Conversion tables	55
BIND control parts	55
Resource associations part	56

Chapter 11. Templates for COBOL

generation	57
Types of templates	58
Preparation templates	58
Preparation JCL for MVS or VSE	62
Preparation CL for OS/400	67
Preparation REXX for VM	70
BIND command templates	71
Link-edit templates	71
Link-edit templates for VM	71
CICS table templates	72
JOB statements	74
Run-time file templates	74
File and database allocation templates	80
File and database allocation placeholder templates	81
Modifying templates	82
Reasons to modify templates	83
Modifying templates and procedures for MVS environments	84
Modifying templates for the OS/400 environment	92
Modifying templates for VSE environments	93
Modifying preparation templates and EXECs for VM environments	97
SQL preprocessing for VM	102
Required SQL/DS VM options	104
Setting additional SQL preprocessor options	104
Setting COBOL run-time options for VM	104
Modifying LE user exits	105
Examples of modifying templates	106
Adding a qualifier to the end user data set names	106
Deleting COBOL source code from the workstation after preparation	113
Deleting COBOL source on an MVS or VSE host	114
Modifying a PSB name to match a batch program name	115
Routing output to a system printer for an MVS/TSO CLIST	116

Suppressing Personal Communications messages during file transfer	117
Creating COBOL compile and link listings for CICS for OS/2	118
Initializing the environment for CICS for OS/2	119
Suppressing CICS translator, COBOL compile, and link messages for CICS for OS/2	119
Processing templates	120
EFK2MPCB before modification	120
Values for the symbolic parameters	120
Preparation JCL created from the EFK2MPCB template	121

Chapter 12. Symbolic parameters 123

Part-related symbolic parameters	123
File-related symbolic parameters	130
User-defined symbolic parameters	131
Creating user-defined symbolic parameters	134
Assigning values to user-defined symbolic parameters	135

Chapter 13. Outputs of COBOL

generation	137
Objects generated for all part types	138
Listing file	141
Preparation script	141
Preparation JCL for MVS and VSE environments	142
Preparation REXX for VM and CICS for OS/2 environments	143
Program and transaction definitions for MVS and VSE environments	143
Objects generated for programs	144
COBOL program	144
Sample run-time CLIST	145
Sample run-time JCL	145
Sample run-time REXX for VM and CICS for OS/2	146
BIND command file	146
Objects generated for map groups	147
Map group format module	148
Batch print services COBOL program	148
Online print services COBOL program	149
MFS print services COBOL program	149
MFS control blocks	149
COBOL copybook for MFS MID/MOD layout	150

Objects generated for tables	150
Table COBOL program	151
Binary table.	151
Modifying generation output	151

Chapter 14. Preparation process for COBOL generation 153

Preparing parts for MVS, VSE, VM, CICS for OS/2 systems	153
Preparation script file contents	154
Additional preparation steps	159
Getting ready for MVS preparation	160
Transferring files to MVS systems	161
Getting ready for VSE preparation	161
Transferring files to VSE systems.	161
Preparing parts for OS/400	161
Preparing parts for CICS for OS/2	162
Analyzing preparation errors	164

Chapter 15. Command interface for COBOL generation 167

GENERATE subcommand syntax for COBOL generation	167
GENERATE subcommand examples	171
PREPARE subcommand syntax for COBOL generation	175
START subcommand syntax	176
STOP subcommand syntax.	177
VALIDATE subcommand syntax for COBOL generation	177

Part 5. Generating Web transaction programs. 179

Chapter 16. Inputs to Web transaction program generation 181

Chapter 17. Outputs of Web transaction program generation 183

Outputs of generating Web transaction program parts	183
Java generation outputs.	183

Chapter 18. Preparation for Web transaction program generation 185

Preparation requirements	185
Preparing Web transaction Java parts	185

Chapter 19. Command interface for Web transaction program generation 187

GENERATE subcommand syntax for Web transaction program generation	187
GENERATE subcommand example for Web transaction	189
Syntax of other HPTCMD subcommands	189

Part 6. Generating JavaBeans wrappers and session beans . . . 191

Chapter 20. Inputs to Java wrapper generation 193

Inputs for JavaBeans wrapper generation	193
Generation options	193
Linkage table options	194

Chapter 21. Outputs of Java wrapper generation 195

Beans for servers	196
Beans for record parameters	198
Beans for record array rows	199

Chapter 22. Command interface for Java wrapper generation 201

GENERATE subcommand syntax for Java wrapper generation	201
START subcommand syntax	202
STOP subcommand syntax.	203
VALIDATE subcommand syntax for Java wrapper generation	204

Chapter 23. Generating session beans 205

Part 7. Reference information 207

Chapter 24. Generation options parts . . . 209

Creating generation options parts	209
Making the default generation options part available during generation	210
Establishing default generation options	210
Default generation options parts with NOOVERRIDE.	210
Default generation option part without NOOVERRIDE.	211
Using multiple levels of generation options parts	212
Determining generation option resolution order	212

Using the sample generation options parts	213	Sample BIND commands	260
Sample generation options default part	213	Binding when the first program uses SQL	260
Sample generation options parts	213	Binding when the first program does not	
Using symbolic parameters in generation		use SQL	262
option specifications	214	Binding packages instead of plans	262
Generation options that are not valid	215	Error return codes on BIND commands	264
Overriding a value to use the default value	215	Binding OS/2 program plans	264
Guidelines for setting generation options	215	Binding for VSE, OS/400, and VM programs	264
Chapter 25. Linkage tables	217	Chapter 28. Resource associations part	265
Creating a linkage table	217	Creating resource associations parts	265
Specifying CALL linkage (CALLLINK)	218	Using multiple resource associations for a	
Definitions for CALLLINK	220	file	266
Valid parameter formats and linkage		Resource association part syntax	266
combinations by platform	231	Sample resource associations part	274
Interfaces requiring a linkage table	232	File types supported by environment and	
Specifying CREATX linkage (CRTXLINK)	234	record organization	274
Specifying DXFR linkage (DXFRLINK)	237	File types supported by CICS	
Interfaces requiring a linkage table	239	environments	276
Specifying File linkage (FILELINK)	239	File types supported for MVS/TSO	286
Sample linkage table entries	242	File types supported for IMS BMP,	
Chapter 26. Link edit parts	245	IMS/VS, and MVS batch	287
Link-editing static COBOL calls	245	File types supported by OS/400	289
Defining a link edit part	245	File types supported for VM CMS and	
Linkage Editor control statements for		VM batch	290
MVS environments	246	File types supported for VSE batch	292
Linkage Editor control statements for VM		Chapter 29. Generation command and	
environments	250	option descriptions	297
Linkage Editor control statements for VSE		HPTCMD commands	297
environments	253	HPTCMD command	297
Specifying AMODE and RMODE	255	HPTCMD subcommands	297
Error return codes on static links	256	Required parameters for subcommands	298
Chapter 27. BIND control parts	257	filename	298
Defining BIND control parts	257	partname	298
Considerations for plan definition	258	/CONFIGMAPNAME	298
Naming CICS for MVS/ESA program		/CONFIGMAPVERSION	299
plans	258	/PROJECT	299
Naming MVS/TSO and MVS batch		/SYSTEM	299
program plans	258	Optional parameters for subcommands	300
Naming IMS program plans	259	/ANSISQL (ANSI SQL statements)	300
Effects of XFER, DXFR, CALL,		/BIND (Bind Control)	301
CONVERSE, and the /RT generation		/CHECKTYPE (Substructured data items)	302
option on plans	259	/CICSDBCS (CICS translator supports	
Using host services CICS for MVS/ESA		DBCS)	303
DBRMs	259	/CICSENTRIES (CICS entries)	303
Using MVS/TSO and MVS batch DBRMs	259	/COMMENTLEVEL (Comment level)	
Using IMS/VS DBRMs	260	(COBOL)	304
Additional BIND command keywords	260	/COMMENTLEVEL (Generate comments)	
		(C++)	305

/CONTABLE (Conversion table)	306	/JOB CARD (JOB card)	323
/CREATEDDS (Create DDS files)	306	/JOBNAME (Job name)	324
/CURRENCY (Currency Symbol)	306	/JSPRELDIR	324
/DATA (Data)	306	/LEFTJUST (Left justify)	324
/DBMS (Database management system)	307	/LINEINFO (Line trace information)	325
/DBPASSWORD (Password)	308	/LINES (Lines per page)	325
/DBUSER (User ID)	308	/LINKAGE (Linkage table)	326
/DEBUGTRACE (Debug trace information)	308	/LINKEDIT (Link edit)	326
/DESTACCOUNT (Account)	309	/LISTING /LISTINGONERROR, /NOLISTING (Generation listing)	326
/DESTDIR (Directory)	309	/LISTINGONERROR	326
/DESTHOST (Name)	310	/LOCVALID (Local data items)	327
/DESTLIB (Target library)	310	/LOG (Log identifier)	327
/DETPASSWORD (Password)	310	/MATH (Math)	327
/DESTUID (User ID)	311	/MFSDEV	328
/DXFRCANCEL (Cancel program after DXFR)	311	/MFSEATTR (MFS extended attribute)	329
/DXFRXCTL (Implement DXFR as an XCTL)	311	/MFSEATTRNCD	329
/EJBGROUP (Enterprise Java Bean Group)	311	/MFSIGNORE (Include IGNORE for SOR)	329
/ENDCOMMAREA (End COMMAREA with FFFF)	312	/MFSTEST (Use test library)	330
/ERRDEST (Error destination)	312	/MSGTABLEPREFIX	330
/FASTPATH (Run as a fast-path program)	313	/MSP (Mapping service program)	330
/FOLD (Fold to uppercase)	313	/NOANSISQL	331
/FTPTRANSLATIONCMDDBCS (FTP DBCS Translation Command)	313	/NOCICSDBCS	331
/FTPTRANSLATIONCMDSBCS (FTP SBCS Translation Command)	314	/NOCREATEDDS	331
/GENAUTHORTIMEVALUES	314	/NODXFRCANCEL	331
/GENHELPMAPS (Help map group)	314	/NODXFRXCTL	331
/GENMAPS (Map group)	315	/NODEBUGTRACE	331
/GENOUT (Generated output directory)	315	/NOENDCOMMAREA	331
/GENPROPERTIES	316	/NOFASTPATH	331
/GENRESOURCEBUNDLE (Generate as resource bundle)	317	/NOFOLD	331
/GENRET (Issue RETURN IMMEDIATE)	318	/NOGENAUTHORTIMEVALUES	332
/GENTABLES (Tables)	319	/NOGENHELPMAPS	332
/GENUIRECORDS	319	/NOGENMAPS	332
/GROUPNAME (Group name)	319	/NOGENPROPERTIES	332
/INEDIT (Input edit)	320	/NOGENRESOURCEBUNDLE	332
/INITADDWS (Initialize additional working storage records)	320	/NOGENRET	332
/INITRECD (Initialize records)	321	/NOGENTABLES	332
/JAVADESTDIR (Java directory)	321	/NOGENUIRECORDS	332
/JAVADESTHOST (Name)	322	/NOINITADDWS	332
/JAVADESTPASSWORD (Password)	322	/NOINITRECD	332
/JAVADESTUID (User ID)	323	/NOLEFTJUST	332
/JAVASYSTEM (Java target system)	323	/NOLINEINFO	332
		/NOLISTING, /LISTING, /LISTINGONERROR	332
		/NOLOCVALID	332
		/NOLOG	332
		/NOMFSEATTR	332
		/NOMFIGNORE	333
		/NOMFSTEST	333

/NONULLFILL	333	/TRACE (Runtime trace)	350
/NONUMOVFL	333	/TRANSFERTYPE (Transfer method)	351
/NOPREP	333	/TRANSID (Transaction IDs)	352
/NOPREPFIL.	333	/TWAOFF (TWA offset).	353
/NORECOVERY	333	/UNLOAD (Unload parts).	353
/NORUNFILE	333	/VALIDMIX (Validate mixed field moves)	354
/NOSETFULL	333	/VMLOADLIB (VM Load Library)	354
/NOSPZERO	333	/LIB (VSE Library)	355
/NOSQLVALID	333	/WORKDB (Work database)	356
/NOSYNCDXFR	333		
/NOSYNCFER	333	Chapter 30. Java properties files	359
/NOUNLOAD.	333	Resource association properties	359
/NOSYCODES	333	Database default properties	361
/NULLFILL (Fill map field)	334	JVM command property	362
/NUMOVFL (Numeric overflow)	334	Java server communication properties	362
/OPTIONS (Generation options).	334	NLS Properties.	364
/PACKAGENAME (Package Name)	335	Linkage properties	364
/POSSIGN (Positive Sign Indicator).	335		
/PREP (Start preparation command file)	335	Chapter 31. Analyzing return codes and errors	367
/PREPFIL. (Create preparation command file)	336	Analyzing generation errors	367
/PRINTDEST (Print destination).	336	Analyzing return codes	367
/PROJECTID (Project ID)	337	Locating generation error messages	367
/RECOVERY (Recover current error message).	337	Analyzing messages	367
/RESOURCE (Resource associations)	338	Analyzing preparation errors	368
/RESOURCEBUNDLELOCALE	339	Analyzing return codes	368
/RESVWORD (Reserved words)	340	Locating preparation error messages	368
/RT (Return or Return transaction ID)	341	Analyzing messages	369
/RUNFILE (Create sample run-time JCL, Create a sample clist, or Create a sample REXX exec)	342		
/SENDTRANSLATIONCMDDBCS (Send DBCS Translation Command).	342	Chapter 32. Serviceability	371
/SESSION (Session ID)	342		
/SETFULL (Set map item FULL).	342	<hr/>	
/SP (Issue CICS SET/INQUIRE).	343	Part 8. Appendixes	373
/SPA (SPA)	343		
/SPZERO (Interpret spaces as zero in NUM and NUMC data items).	343	Appendix A. List of valid generation options for each environment	375
/SQLDB (SQL database)	344		
/SQLID (SQL userid)	345	Appendix B. Implementing a generation server	381
/SQLPASSWORD (password).	345	Setting up the client	382
/SQLVALID (SQL statements).	345	Setting up the server.	383
/SYMPARM	346	Setting up Object REXX support on Windows NT	384
/SYNCDXFR (Set sync points for DXFRs)	347	LAN generation setup	385
/SYNCFER (Set sync points for XFRs)	347	Setup on the generation server	385
/SYSCODES (Use system return codes)	348	Setup on the client	386
/TARGNLS (Target NLS)	348		
/TEMPLATES (Templates directory)	350	Appendix C. Reading syntax diagrams	389
		Index	391

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood NY 10594, U.S.A.

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 the SWS General Legal Counsel, IBM Corporation, Department TL3 Building 062, P. O. Box 12195, Research Triangle Park, NC 27709-2195. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

IBM has made reasonable efforts to ensure the accuracy of the information contained in this publication. If a softcopy of this publication is provided to you with the product, you should consider the information contained in the softcopy version the most recent and most accurate. However, this publication is presented "as is" and IBM makes no warranties of any kind with respect to the contents hereof, the products listed herein, or the completeness or accuracy of this publication.

IBM may change this publication, the product described herein, or both.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries:

- ACF/VTAM
- AD/Cycle
- AIX
- AS/400
- C Set ++
- CICS
- CICS OS/2
- CICS/ESA
- CICS/MVS
- CICS/VM
- CICS/VSE
- CICS/400
- CICS/6000
- COBOL/2
- COBOL/400
- DB2
- IBM
- IBMLink
- IMS
- IMS/ESA
- Language Environment
- MVS
- MVS/ESA
- Operating System/2
- OS/2
- OS/400
- SAA
- SQL/DS
- SQL/400
- VisualAge
- VisualGen
- VM/ESA
- VSE/ESA
- WebSphere

The following are trademarks of other companies:

C++

American Telephone & Telegraph
Company

HP-UX	Hewlett-Packard Company
Oracle	Oracle Corporation

Microsoft, Windows, Windows NT, the Windows 95 logo, and the Windows 98 logo are trademarks or registered trademarks of Microsoft Corporation.

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

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be the trademarks or service marks of others.

Terminology used in this document

Unless otherwise noted in this publication, the following references apply:

- MVS CICS applies to Customer Information Control System/Enterprise Systems Architecture (CICS/ESA) systems.
- CICS applies to CICS for VSE/ESA, CICS/ESA, CICS for OS/2, CICS for AIX, CICS for Windows NT, and CICS for Solaris.
- CICS for Windows NT refers to IBM TXSeries for Windows NT Version 4.2.
- CICS for AIX refers to IBM TXSeries for AIX Version 4.2.
- CICS for Solaris refers to IBM WebSphere Enterprise Edition Version 3.0.
- IMS/VS applies to Information Management System/Enterprise System Architecture (IMS/ESA) and IMS/ESA Transaction Manager systems.
- IMS applies to IMS/ESA and IMS/ESA Transaction Manager, and to message processing program (MPP), IMS Fast Path (IFP), and batch message processing (BMP) regions. IMS/VS is used to distinguish MPP and IFP regions from the IMS BMP target environment.
- LE applies to the IBM Language Environment for MVS and VM.
- COBOL applies to any of the following types of COBOL:
 - IBM VisualAge for COBOL for OS/2
 - ILE COBOL/400
 - IBM COBOL for VSE
 - IBM COBOL for MVS and VM
- “Region” and “CICS region” correspond to the following:
 - CICS for MVS/ESA region
 - IMS region
 - CICS for VSE/ESA partition
 - CICS for OS/2 system
 - CICS for AIX system
 - CICS for Windows NT system
 - CICS for Solaris system
- DB2/VSE refers to SQL/DS Version 3 Release 4 or later. Any references to SQL/DS refer to DB2/VSE and SQL/DS on VM. In addition, any references to SQL/400 refer to DB2/400.
- OS/2 CICS applies to CICS Operating System/2 (CICS for OS/2).
- Workstation applies to a personal computer, not an AIX workstation.
- The make process applies to the generic process not to specific make commands, such as make, nmake, pmake, polymake.
- Unless otherwise noted, references to VM apply to Virtual Machine/Enterprise Systems Architecture (VM/ESA) environments.

- References to VM batch apply to any batch facility running on VM.
- DB2/2 applies to DB2/2 Version 2.1 or later, and DB2 Universal Database (UDB) for OS/2 Version 5.
- DB2/6000 applies to DB2/6000 Version 2.1 or later, and DB2 Universal Database (UDB) for AIX Version 5.
- Windows applies to Windows 95, Windows 98, Windows NT, and Windows 2000.
- Unless a specific version of Windows NT is referenced, statements regarding Windows NT also apply to Windows 2000.

Terminology differences between Java and Smalltalk

VisualAge Generator Developer can be installed as a feature of VisualAge for Java or VisualAge Smalltalk. Where appropriate, the documentation uses terminology that is specific to Java or Smalltalk. But where the information is specific to VisualAge Generator and virtually the same for both environments, the Java/Smalltalk term is used.

Table 1. Terminology differences between Java and Smalltalk

Java term	Combined Java/Smalltalk term	Smalltalk term
Project	Project/Configuration map	Configuration map
Package	Package/Application	Application
Workspace	Workspace/Image	Image
Beans palette	Beans/Parts palette	Parts palette
Bean	Visual part or bean	Visual part
Repository	Repository/ENVY library	ENVY library manager
Options	Options/Preferences	Preferences

About this document

This document provides information needed to prepare and generate programs for the AIX, HP-UX, SCO OpenServer, Solaris, Microsoft Windows, MVS, OS/2, OS/400, VM, and VSE environments using VisualAge Generator.

This document contains the following information:

- An introduction to the generation processes, including the following:
 - The target environments
 - The VisualAge Generator generation process to prepare COBOL, C++, and Java programs
- Sections about using VisualAge Generator to generate COBOL, C++, Java, Web transaction, and JavaBeans wrapper programs
- A section containing information that applies to all types of programs

For a list of related publications, see “Documentation provided with VisualAge Generator”. Refer to the appropriate document for information about running generated programs for your target environment.

For information about client generation and generating VAGen run-time code, see the *VisualAge Generator User's Guide*.

How to use this document

This document describes the tasks that are required to generate VisualAge Generator source into an executable format. The outputs from generation vary depending on the target environment. The following types of output can be produced:

- COBOL
- C++
- Java

You should read the sections of this book that correspond to the type of output.

Documentation provided with VisualAge Generator

VisualAge Generator documents are provided in one or more of the following formats:

- Printed and separately ordered using the individual form number.

- Online book files (.pdf) on the product CD-ROM. Adobe Acrobat Reader is used to view the manuals online and to print desired pages.
- HTML files (.htm) on the product CD-ROM and from the VisualAge Generator web page (<http://www.ibm.com/software/ad/visgen>).

The following books are shipped with the VisualAge Generator Developer CD. Updates are available from the VisualAge Generator Web page.

- *VisualAge Generator Getting Started* (GH23-0258-01) ^{1,2}
- *VisualAge Generator Installation Guide* (GH23-0257-01) ^{1,2}
- *Introducing VisualAge Generator Templates* (GH23-0272-01) ^{2,3}

The following books are shipped in PDF and HTML formats on the VisualAge Generator CD. Updates are available from the VisualAge Generator Web page. Selected books are available in print as indicated.

- *VisualAge Generator Client/Server Communications Guide* (SH23-0261-01) ^{1, 2}
- *VisualAge Generator Design Guide* (SH23-0264-00) ¹
- *VisualAge Generator Generation Guide* (SH23-0263-01) ¹
- *VisualAge Generator Messages and Problem Determination Guide* (GH23-0260-01) ¹
- *VisualAge Generator Programmer's Reference* (SH23-0262-01) ¹
- *VisualAge Generator Migration Guide* (SH23-0267-00) ¹
- *VisualAge Generator Server Guide for Workstation Platforms* (SH23-0266-01) ^{1,4}
- *VisualAge Generator System Development Guide* (SG24-5467-00) ²
- *VisualAge Generator User's Guide* (SH23-0268-01) ^{1, 2}
- *VisualAge Generator Web Transaction Development Guide* (SH23-0281-00) ¹

The following documents are available in printed form for VisualAge Generator Server for AS/400 and VisualAge Generator Server for MVS, VSE, and VM:

- *VisualAge Generator Server Guide for AS/400* (SH23-0280-00) ²
- *VisualAge Generator Server Guide for MVS, VSE, and VM* (SH23-0256-00) ²

The following information is also available for VisualAge Generator:

- *VisualAge Generator External Source Format Reference* (SH23-0265-01)
- *Migrating Cross System Product Applications to VisualAge Generator* (SH23-0244-01)
- *VisualAge Generator Templates V4.5 Standard Functions—User's Guide* (SH23-0269-01) ^{2, 3}

1. These documents are available as HTML files and PDF files on the product CD.

2. These documents are available in hardcopy format.

3. These documents are available as PDF files on the product CD.

4. This document is included when you order the VisualAge Generator Server product CD.

Part 1. Introducing generation

Chapter 1. Introducing program generation

The generation process uses VisualAge Generator (VAGen) part definitions, which are created using VisualAge Generator Developer and are stored in the repository. The part definitions and control files are used to generate COBOL or C++ programs, depending upon the target environment.

You can generate programs, tables, and maps from the user interface or from the command line using the HPTCMD command.

Figure 1 shows the Generate window where you can select the target system.

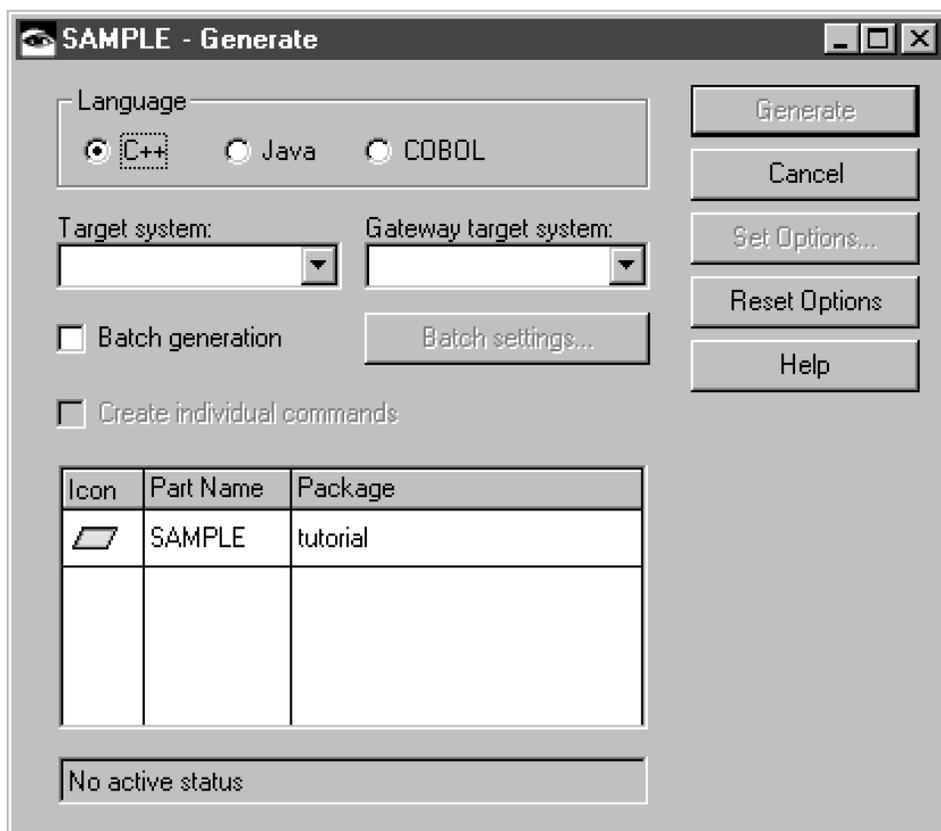


Figure 1. Generate package/application

The generation options part enables you to customize the generation process for specific target environments. You can specify generation options using the following methods:

- By entering values using the VisualAge Generator Developer user interface
- As options on the command interface when you are using the GENERATE subcommand
- By setting values for these options in generation options parts

Figure 2 shows the Generation Options window where you set generation options such as validation parameters.

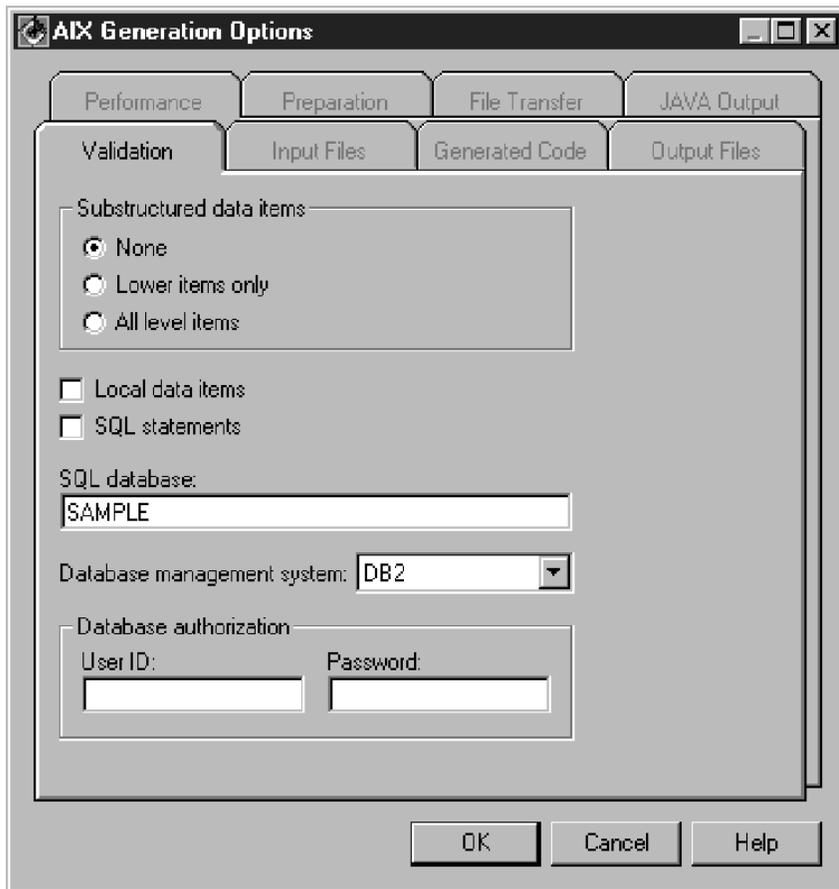


Figure 2. Generation Options

Generation options are stored in generation options parts, which enable developers working on the same project to share generation options. You can specify a default generation options part using the /OPTIONS option on the

GENERATE subcommand. A generation options part can also specify the /OPTIONS option, creating a succession of generation options parts.

For C++, Java, and COBOL programs, the generation process consists of the following processes:

Generation

Generation includes the following steps:

Validation

Collects definition information from parts for the object being generated. Validation also includes cross-checking the parts to ensure that the definitions are complete and correct. Information and error messages are issued when problems are detected. The generation process stops if there are errors. You can validate the parts without generating the program.

Production

Builds the source code and related program objects from the part definitions during this phase.

Preparation

Preparation includes the following steps:

- Transferring parts to the target environment
- Running precompilers, compilers, and linkers on the target system

The generation and preparation processes work together. On the development system, generation produces source code and related program objects. Then, on the target system, the preparation process prepares the generated source to run.

Generating C++, Java, and COBOL programs

The generation process transforms programs, maps, map groups, and tables into source code and related program objects. Outputs are created based on the selected generation options and target environment.

During generation, the VAGen part definitions are transformed into C++, Java, or COBOL programs and related program objects. Outputs are created based on the generation options selected.

You can generate a program using the batch command interface or the user interface. Refer to the VisualAge Generator Developer online help system for information about the user interface.

The VisualAge Generator Developer command interface syntax is specified by the command HPTCMD, followed by a subcommand. The VALIDATE

subcommand specifies that you want validation only, and the GENERATE subcommand specifies that you want both validation and production. Using the VisualAge Generator Developer command interface, you can issue HPTCMD commands one at a time by specifying them on the command line, or you can create a command file to issue multiple HPTCMD commands.

The generation process produces several output files, some of which can be modified. The outputs of generation must be transferred to the target system and prepared (for example, compiled and linked) on the target system before they can be used.

Preparing C++, Java, and COBOL programs

After generating a program, you must prepare it for the run time environment. The preparation process uses the outputs from the generation process to prepare the source objects for run time.

Preparation can be started as part of the generation process or run independently. The preparation process is automatically started by generation unless you specify the /NOPREP option on the GENERATE subcommand.

To run the preparation process independently of the generation process, specify the PREPARE subcommand. The PREPARE subcommand can only be issued through the command interface.

The steps involved in the preparation process differ according to the target system.

- If the development and target system are the same OS/2 system, the preparation process performs the compile and link steps. If the target system is a remote OS/2 system, the preparation process transfers the source objects from the source to the OS/2 target system and begins the compile and link process on the target system.
- If the development and target system are the same Windows NT system, the preparation process performs the compile and link steps. If the target is a remote Windows NT system, the preparation process transfers the source objects from the source machine to the Windows NT target machine, but the compile and link steps cannot be started automatically. The compile and link steps must be started manually by invoking the preparation command on the Windows NT target machine.
- If the target system is MVS, VM, VSE, or OS/400, the preparation process transfers the source objects from the workstation to the target system and begins the compile and link process on the target system.
- If the target system is AIX, HP-UX, SCO, or Solaris, the preparation process transfers the source objects from the development machine to the AIX,

HP-UX, SCO, or Solaris target machine and compiles and links on the AIX,
HP-UX, SCO, or Solaris target machine.

Part 2. Generating C++ programs

Chapter 2. Inputs to C++ program generation

The C++ program generation process uses inputs from several different sources:

- VisualAge Generator part definitions that define the program, map group, or table, and its associates.
Refer to the VisualAge Generator Developer online help system for more information about defining parts.
- Generation control files and parts, which are predefined. The generation control files and parts for C++ are described in the following sections:
 - “Linkage table parts” on page 53
Refer to the *VisualAge Generator Client/Server Communications Guide* document for information about linkage tables.
 - “Chapter 24. Generation options parts” on page 209
 - The resource associations part is used as an input of generation when the target environment is CICS for AIX, CICS for Solaris, or CICS for Windows NT.

These control parts and files contain installation and project-level conventions that specify how the C++ programs and related objects are generated. Generation control parts and files are usually specified at an installation or project level, but they can be specified at any level. You can modify the contents of the generation control parts and files.

Note: The resource associations part is used during the generation process for C++ programs only when the target environment is CICS for AIX, CICS for Solaris, or CICS for Windows NT. C++ programs running on OS/2, AIX, HP-UX, SCO, Solaris, and Windows NT use resource association files at run time. The contents of the resource association file indicate where the physical files used by the program are located.

Refer to the *VisualAge Generator Server Guide for Workstation Platforms* document for information about the resource association file and supported file types for the OS/2, AIX, HP-UX, SCO, Solaris, and Windows NT environments.

Inputs for C++ generation

The following tables list the inputs for C++ generation.

Table 2 lists the C++ inputs for the supported environments.

Table 2. Inputs for C++ generation

Element	File name and extension	Environment	Description
Part definition	Stored in repository	All	
Environment variables	Set in the <i>hpt.ini</i>	All	Variables that can affect the generation process
User-defined option files	Stored in repository	All	
Linkage table	Stored in repository	All	Optional table
Conversion tables	ELAxxxxx	AIX, CICS for AIX, HP-UX, SCO, Solaris, CICS for Solaris, OS/2, Windows NT, CICS for Windows NT	
Resource association part	Stored in repository	CICS for AIX, CICS for Solaris, CICS for Windows NT	

Generation options parts

C++ generation uses generation options parts. See “Chapter 24. Generation options parts” on page 209 for information about generation options parts.

Linkage table parts

Linkage table parts are required if the program contains external calls to non-VisualAge Generator programs. A linkage table specifies the following:

- The linkage conventions to be used for calling a program
- Whether a CICS CREATX service call starts a local or remote CICS transaction
- The linkage conventions to be used for implementing a DXFR transfer between host programs
- Whether a CICS file is to be accessed as a local or remote file

Refer to the *VisualAge Generator Client/Server Communications Guide* for information about linkage tables.

Resource associations part

The resource associations part is used to generate C++ programs for CICS for AIX, CICS for Solaris, and CICS for Windows NT.

See “Chapter 28. Resource associations part” on page 265 for details.

Note: The resource association part is used during the generation process for C++ programs only when the target environment is CICS for AIX, CICS for Solaris, or CICS for Windows NT. When the target environment is OS/2, AIX, HP-UX, SCO, Solaris, or Windows NT, C++ programs use resource association files at run time. The contents of the resource association file indicate where the physical files used by the program are located.

Refer to the *VisualAge Generator Server Guide for Workstation Platforms* for information about the resource association file and supported file types for the OS/2, AIX, HP-UX, SCO, Solaris, and Windows NT environments.

Conversion tables

Conversion tables are used to convert data across OS/2 ASCII, Windows ASCII, UNIX ASCII, and EBCDIC systems when generating binary objects (tables). Conversion tables are specified using the /CONTABLE generation option. See “Chapter 29. Generation command and option descriptions” on page 297 for more information about the /CONTABLE generation option and about conversion tables.

Refer to the *VisualAge Generator Client/Server Communications Guide* for information on how to define a custom conversion table for use with code pages associated with other languages for client/server programs.

Chapter 3. Outputs of C++ program generation

The outputs of the generation process are C++ source and related objects needed to prepare and run your program. These outputs contain the information necessary to transfer files and start the appropriate compile and link processes.

If you are generating a Web transaction program, output includes Java source and related objects. For more information on the generated files, refer to “Chapter 17. Outputs of Web transaction program generation” on page 183.

Depending on whether the program uses SQL or not, certain interim preparation steps occur. Table 3 include the preparation steps and the outputs of generation.

Table 3. Generation outputs for AIX, CICS for AIX, HP-UX, SCO OpenServer, Solaris, CICS for Solaris, OS/2, Windows NT, and CICS for Windows NT

Element	File name and extension	Uploaded	Environment
C++ source for each map group and program generated	xxxxxxx.CPP ¹	Yes	All
Binary source for each table generated	xxxxxxx.TAB	Yes	All
Preparation file for each program generated	xxxxxxx.CMD ²	No	All
Command file for each program generated; starts the make file	xxxxxxxZ.CMD	Yes	OS/2
File transfer protocol (FTP) control file	xxxxxxx.FTP	No	AIX, CICS for AIX, HP-UX, SCO, Solaris, CICS for Solaris, Windows NT, CICS for Windows NT, and OS/2 if development and target systems differ
UNIX script file containing the make process used on the remote machine	xxxxxxxZ.scr	Yes	AIX, CICS for AIX, HP-UX, SCO, Solaris, and CICS for Solaris
UNIX script file for CICS program and transaction definitions	xxxxxxxC.scr	Yes	CICS for AIX and CICS for Solaris

Table 3. Generation outputs for AIX, CICS for AIX, HP-UX, SCO OpenServer, Solaris, CICS for Solaris, OS/2, Windows NT, and CICS for Windows NT (continued)

Element	File name and extension	Uploaded	Environment
Command file for each program generated; starts the make file	xxxxxxxZ.BAT	Yes	Windows NT, CICS for Windows NT
Command file containing CICS program and transaction definitions	xxxxxxx.C.BAT	Yes	CICS for Windows NT
Dynamic link library / Shared library files	xxxxxxx.dll	Yes	AIX, OS/2, Windows NT
	xxxxxxx.ibmcpp	Yes	CICS for Windows NT and CICS for AIX
Shared library files	xxxxxxx.sl	Yes	HP-UX
	xxxxxxx.so	Yes	SCO and Solaris
	xxxxxxx.cpp	Yes	CICS for Solaris

Notes:

¹xxxxxxx indicates a 7-character file name.

²xxxxxxxZ indicates an 8-character file name with Z required as the last character of the name.

Generated objects are stored in the directory specified on the /GENOUT generation option. If /GENOUT specifies a location that is not valid or if /GENOUT is not specified, the generation objects are stored in the directory where the server process is running.

Chapter 4. Preparation process for C++ generation

When generation is completed, the generated outputs must be prepared for run time, just as you prepare programs you might write yourself.

Preparation takes place automatically unless you specify the `/NOPREP` option. If you specify the `/NOPREP` option, preparation takes place as a separate process. The `/PREP` option is the default.

The preparation process includes the following steps:

- Transferring parts to the target environment, if needed
- Unicode conversion, if needed
- Running precompilers, compilers, and linkers

When you specify the `/PREPFILE` generation option without the `/PREP` generation option, the `PREPARE` subcommand can be used to prepare the generation output.

The `PREPARE` subcommand can also be used to restart preparation if the preparation process is not successful in a manner that does not require the parts be generated again. The `PREPARE` subcommand is used for preparing programs generated using any VisualAge Generator Developer.

A `.CMD` file is generated for each program, table, or map group being generated. The `.CMD` file is run unless the `/NOPREP` option is specified. The `.CMD` file uses the `FCEBUILD.EXE` to control preparation processing.

See “Chapter 31. Analyzing return codes and errors” on page 367 for information about preparation errors.

Preparing C++ programs

The `FCEBUILD.EXE` does the following processing:

1. If the development machine is different from the target machine, the `pgm.FTP` file is used to control transfer of all the required files to the target machine. This transfer includes the `pgmZ.scr` for AIX, CICS for AIX, HP-UX, SCO, Solaris, and CICS for Solaris, the `pgmZ.CMD` file for OS/2, and the `pgmZ.BAT` file for Windows NT and CICS for Windows NT. The transfer also includes the generated outputs of any tables, maps, and map groups used by the program.
2. To start the make process, the `pgmZ.scr` is run on the AIX, CICS for AIX, HP-UX, SCO, Solaris, or CICS for Solaris machine, the `pgmZ.CMD` is run

on the OS/2 machine, or the pgmZ.BAT is run on a local Windows NT or CICS for Windows NT machine. The make process includes the following steps:

- a. Precompiles each SQL program
- b. Compiles each program and map group
- c. Links the files to create the dynamic link library or shared library

If the target machine is a remote Windows NT or CICS for Windows NT machine, the pgmZ.BAT must be run manually to start the make process.

For non-CICS environments, use the FCWRUN.EXE file to run the dynamic link library or shared library that is created.

When the target environment is CICS for AIX, CICS for Solaris, or CICS for Windows NT, the program transaction files must be defined to CICS for AIX, CICS for Solaris, or CICS for Windows NT before the program can run. Refer to the *VisualAge Generator Server Guide for Workstation Platforms* for information on how set up your CICS for AIX, CICS for Solaris, or CICS for Windows NT system and how to run your prepared program.

Analyzing preparation errors

When you analyze the preparation errors for the OS/2, AIX, HP-UX, SCO, Solaris, or Windows NT environments, do the following:

1. Verify that the pgmZ.CMD, pgmZ.BAT, or pgmZ.SCR file contains the correct compile information for your program.
2. Use the SET command to display information about environment variables settings for an OS/2, AIX, HP-UX, SCO, Solaris, or Windows NT session.

Common preparation errors

The most common errors during preparation are these:

- Errors accessing the database
- FCWMAKE not found
- The preparation jobs do not run on the AIX, HP-UX, SCO, or Solaris target.

Errors accessing the database

The database on your development system might not match the database you are using on the target system. The default database name might be different, or the USERID and passwords might be different. Modify your environment variables or generation options to use the correct values for your target environment.

FCWMAKE not found

The example below shows which SET statement in your CONFIG.SYS file to check if the preparation job cannot find FCWMAKE.

```
set fcwmake=directory
```

Where *directory* is the location of VisualAge Generator Server for OS/2 or VisualAge Generator Server for Windows NT.

Preparation jobs do not run on the target system

Preparation jobs cannot run successfully on the target system when any of the following requirements are not satisfied:

- Because the AIX, HP-UX, SCO, and Solaris environments are case sensitive, be sure to verify the case for the specified values for the following options:
 - /DBPASSWORD
 - /DBUSER
 - /DESTDIR
 - /DESTHOST
 - /DESTUID
 - /DESPASSWORD
- The user specified on the /DESTUID option must have read and execute authority to the target machine (specified by the /DESTHOST option) and must also have write authority to the target directory (specified by the /DESTDIR option).

The example below shows the command you type at a command prompt where you are running the generator to check for read and execute authority.

```
rexec <desthost> -l <destuid> -p <destpassword> df
```

You can check for write authority by doing the following:

1. Log on to the AIX, HP-UX, SCO, or Solaris machine using the values you are specifying for the /DESTUID and /DESPASSWORD options.
2. Change to the directory you are specifying on the /DESTDIR option using the cd command.
3. Test your access by creating a file.

The example below shows an example command that lists the files and redirects the list to a file. This is one way to create a file.

```
ls -la > temp.file
```

If you are able to create the file, you have write authority.

Note: You can specify relative or full path names for the /DESTDIR option, but fully qualified path names are recommended.

- Verify the case for the values assigned to the *sqlDefaultDatabase* key in the HPT.INI file.
- If your program uses SQL, the database application development software (DB2 SDK for DB2 or Oracle Programmer/2000 for Oracle) must be installed on the preparation machine.

- The correct value must be set for SQL validation and the correct database must be used.

Either of the following combinations are valid:

- If you have DB2 installed on your development system, but are generating and preparing a job for an AIX, HP-UX, SCO, or Solaris target system, you must specify the /NOSQLVALID generation option.
 - If you have DDCS installed on your development system, but are generating and preparing a job for an AIX, HP-UX, SCO, or Solaris target system, you can specify the /SQLVALID generation option. You must define the AIX, HP-UX, SCO, or Solaris database to DB2 as a remote database.
- Preparation for C++ for OS/2 requires OS/2 Warp 4.0 or later.
 - Tables must all be specified in lowercase. For dynamic link libraries (DLLs) or shared libraries, the file name must be all uppercase, but the file extension must be lowercase. The example below shows some examples for file names for tables and DLLs.

```
mytable.tab  
MYAPP.d11  
MYMAPS.d11
```

If you use your own preparation process, be sure to name the files correctly. If you use the preparation process provided with VisualAge Generator, it moves the files to the target environment. You must be sure to name the files correctly. If you do not specify the /NOPREP generation option, the VisualAge Generator Developer moves the files for you and correctly names them.

Chapter 5. Command interface for C++ generation

You can issue the VisualAge Generator Developer subcommands from a system prompt or from within a command file. The command HPTCMD implements the command interface. Subcommands are specified with the HPTCMD command and are followed by any required keywords and options. Comments can be imbedded in the commands. Comments begin with the characters `/*` and end with the characters `*/`.

Command processing can be started explicitly by issuing the START subcommand or implicitly by issuing any other VisualAge Generator Developer subcommand. The command continues running until it is ended, either by issuing the STOP subcommand or by closing the Generation Monitor window.

Starting the command opens a Generation Monitor window. The Generation Monitor window displays the command currently being processed and provides information showing what stage of generation the process has reached. You can cancel the currently processing command from the Generation Monitor window. Closing the Generation Monitor window ends any command currently being processed.

If you are generating programs using a generation server, refer to the *VisualAge Generator System Development Guide* for information about starting and stopping the Generation Monitor.

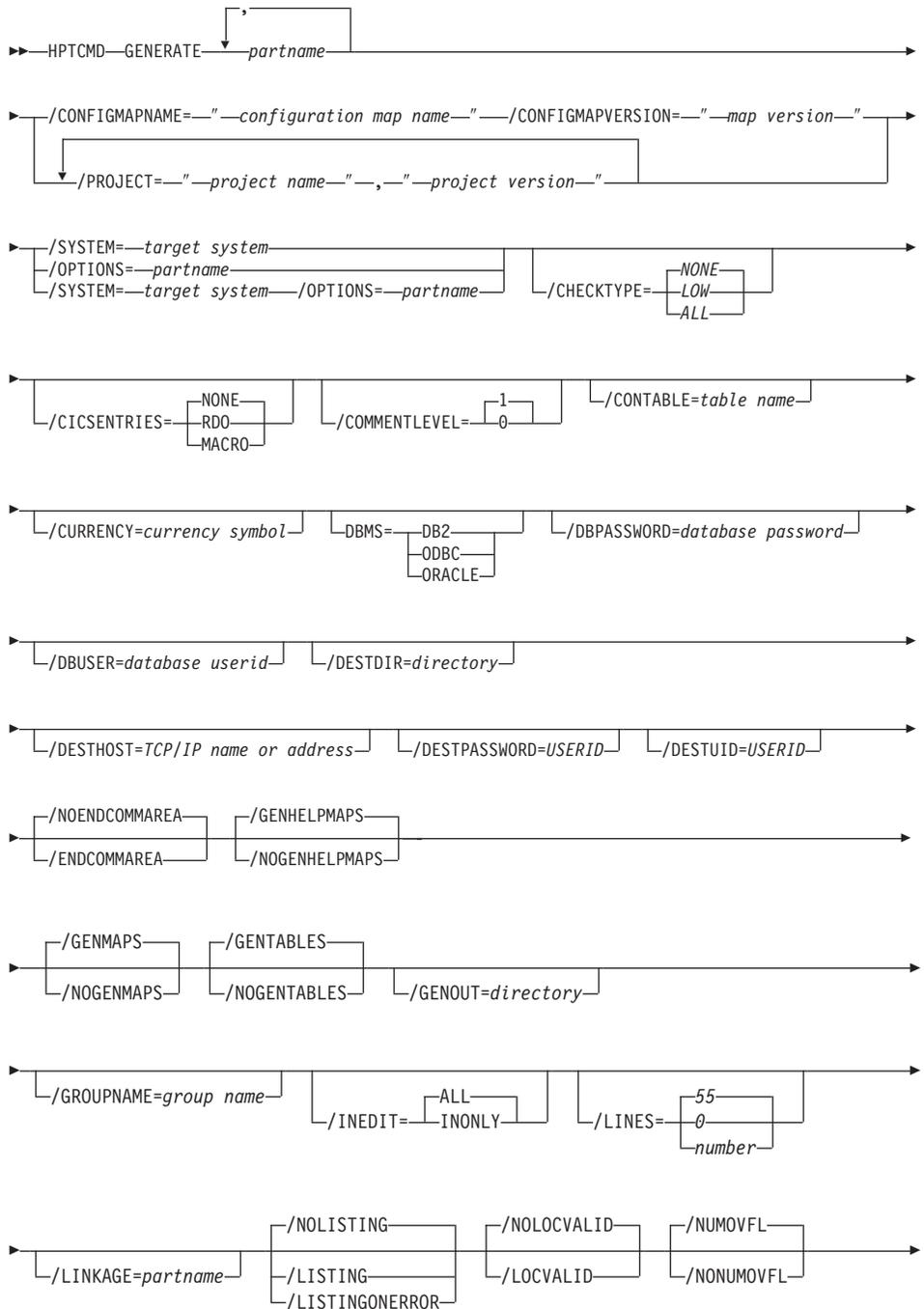
Note: If you are generating a program that uses DBCS, you must run the commands on a machine that is DBCS-enabled.

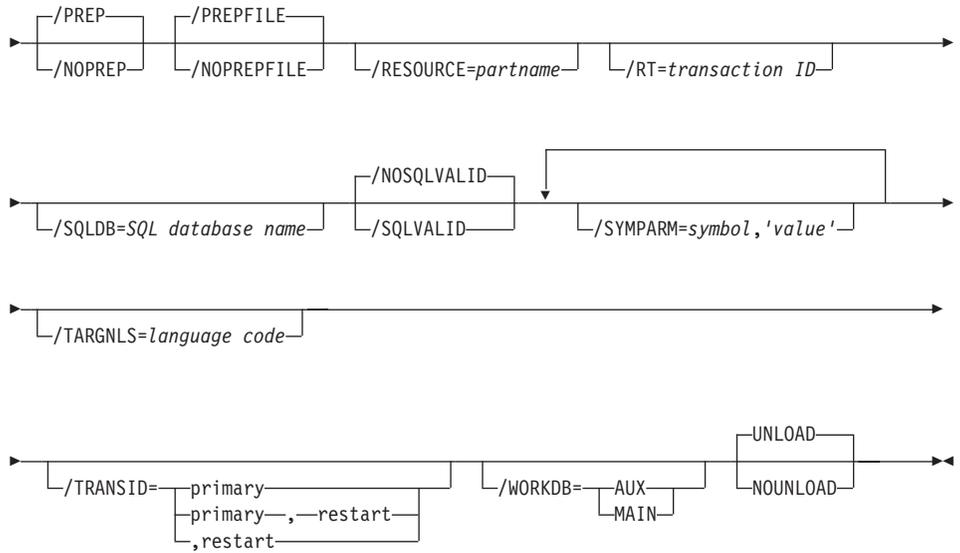
GENERATE subcommand syntax for C++ generation

For C++ generation, the GENERATE subcommand enables you to generate a program, table, or map group. You can also specify options that affect how a part is generated.

The following syntax diagram shows the options required for the GENERATE subcommand.

Note: The `/CONFIGMAPNAME`, `/CONFIGMAPVERSION`, and `/SYSTEM` options require a value when you generate from the command interface by using VisualAge Generator on Smalltalk. The `/PROJECT` and `/SYSTEM` options require a value when you generate from the command interface using VisualAge Generator on Java.





See “Chapter 29. Generation command and option descriptions” on page 297 for the detailed information about the syntax diagram.

Example

Figure 3 illustrates the GENERATE subcommand for C++ program generation; at an OS/2 command prompt it must be entered in one continuous line. If you are using VisualAge Generator on Java, use /PROJECT instead of /CONFIGMAPNAME and /CONFIGMAPVERSION.

```
HPTCMD GENERATE MYPGM
/CONFIGMAPNAME="myconfig" /CONFIGMAPVERSION="27.8"
/SYSTEM=AIX
/COMMENTLEVEL=0
/DESTDIR='/u/pancho'
/DESTHOST=quixote.cary.ibm.com
/DETPASSWORD=mypassword
/DESTUID=pancho
/OPTIONS=MYOPT
/GENOUT=D:\pancho\gen /NUMOVFL
/TRACE=PROCESSES
```

Figure 3. GENERATE subcommand for C++

The AIX, HP-UX, and Solaris environments are case sensitive. Verify the case for the specified values for the following options:

```
/DBPASSWORD
/DBUSER
```

```
/DESTDIR  
/DESTHOST  
/DESTUID  
/DETPASSWORD
```

See “Chapter 24. Generation options parts” on page 209 for information about setting generation options.

See “Chapter 31. Analyzing return codes and errors” on page 367 for information about generation return codes.

GENERATE subcommand examples for C++

Each example in this section illustrates different uses of the GENERATE subcommand and includes the GENERATE syntax that is used. The examples display as several lines because of the formatting limitations of this document, but they must be entered as one command line.

Generating for AIX

Figure 4 shows an example that generates a program for use in the AIX environment.

```
HPTCMD GENERATE PRGM1 /CONFIGMAPNAME="myconfig" /CONFIGMAPVERSION="27.8"  
/SYSTEM=AIX  
/LINKAGE=linktbl  
/DESTDIR='/u/sleepy/genout'  
/DESTHOST=xmaster  
/DESTUID=sleepy  
/DETPASSWORD=secret
```

Figure 4. Generating for AIX

Generating for HP-UX

Figure 5 shows as example that generates a program for use in the HP-UX environment.

```
HPTCMD GENERATE PRGM1 /CONFIGMAPNAME="myconfig" /CONFIGMAPVERSION="27.8"  
/SYSTEM=HP  
/LINKAGE=linktbl  
/DESTDIR='/u/sleepy/genout'  
/DESTHOST=xmaster  
/DESTUID=sleepy  
/DETPASSWORD=secret
```

Figure 5. Generating for HP-UX

Generating for SCO OpenServer

Figure 6 shows an example that generates a program for use in the SCO environment.

```
HPTCMD GENERATE PRGM1 /CONFIGMAPNAME="myconfig" /CONFIGMAPVERSION="27.8"  
/SYSTEM=SCO  
/LINKAGE=linktbl  
/DESTDIR='/u/sleepy/genout'  
/DESTHOST=xmaster  
/DESTUID=sleepy  
/DETPASSWORD=secret
```

Figure 6. Generating for SCO OpenServer

Generating for Solaris

Figure 7 shows an example that generates a program for use in the Solaris environment.

```
HPTCMD GENERATE PRGM1 /CONFIGMAPNAME="myconfig" /CONFIGMAPVERSION="27.8"  
/SYSTEM=SOLARIS  
/LINKAGE=linktbl  
/DESTDIR='/u/sleepy/genout'  
/DESTHOST=xmaster  
/DESTUID=sleepy  
/DETPASSWORD=secret
```

Figure 7. Generating for Solaris

Generating for CICS for AIX

Figure 8 shows an example that generates a program for use in the CICS for AIX environment.

```
HPTCMD GENERATE PRGM1 /CONFIGMAPNAME="myconfig" /CONFIGMAPVERSION="27.8"  
/SYSTEM=AIXCICS  
/LINKAGE=linktbl  
/RESOURCE=fcwrs  
/DESTDIR='/u/sleepy/genout'  
/DESTHOST=xmaster  
/DESTUID=sleepy  
/DETPASSWORD=secret
```

Figure 8. Generating for CICS for AIX

Generating for OS/2

Figure 9 on page 26 shows an example that generates a program for use in the OS/2 environment.

```
HPTCMD GENERATE PRGM1 /CONFIGMAPNAME="myconfig" /CONFIGMAPVERSION="27.8"  
/SYSTEM=OS2  
/LINKAGE=linktbl
```

Figure 9. Generating for OS/2

Generating for Windows NT

Figure 10 shows an example that generates a program for use in the Windows NT environment.

```
HPTCMD GENERATE PRGM1 /PROJECT="myproject", "27.8"  
/SYSTEM=WINNT  
/LINKAGE=linktbl  
/DESTDIR=c:\vggenout  
/DESTHOST=nthost1  
/DESTUID=sleepy  
/DETPASSWORD=secret
```

Figure 10. Generating for Windows NT

START subcommand syntax

The START subcommand starts the server process that runs HPTCMD subcommands.

►—HPTCMD—START—◄

See “Chapter 29. Generation command and option descriptions” on page 297 for the detailed information about the syntax diagram.

If you do not issue the START subcommand and the server process is not already running, any HPTCMD subcommand you issue, except for the STOP subcommand, starts the server process automatically.

Note: If you do not issue the START subcommand and you direct your output to a location other than STDOUT, the server process inherits the STDOUT location. This causes the STDOUT file to be locked by the server until an HPTCMD STOP command is issued. To avoid this, always issue an HPTCMD START command, either at the command line or by placing the command at the beginning of your command file.

Example

The following is an example of how to use the START subcommand:

STOP subcommand syntax

The STOP subcommand stops the server process that runs HPTCMD subcommands.

▶▶—HPTCMD—STOP—▶▶

See “Chapter 29. Generation command and option descriptions” on page 297 for the detailed information about the syntax diagram.

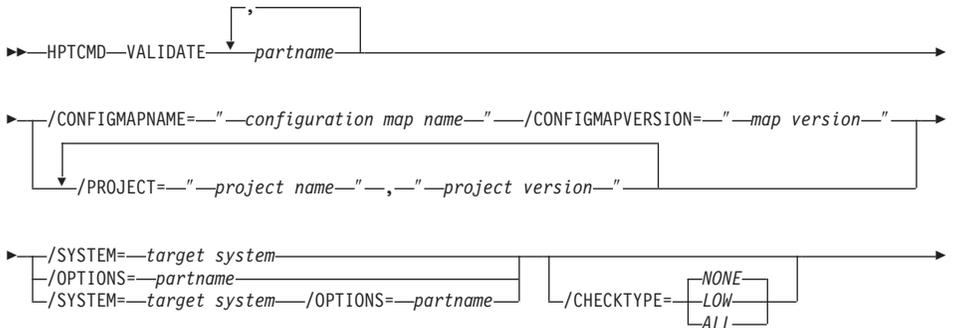
The server process continues to run until it receives a STOP subcommand or until the Generation Monitor window is closed. When you specify the STOP subcommand, the server process completes all previously issued subcommands before stopping. If the server process is stopped as a result of the Generation Monitor window closing, the stop occurs immediately, ending any subcommand currently being processed.

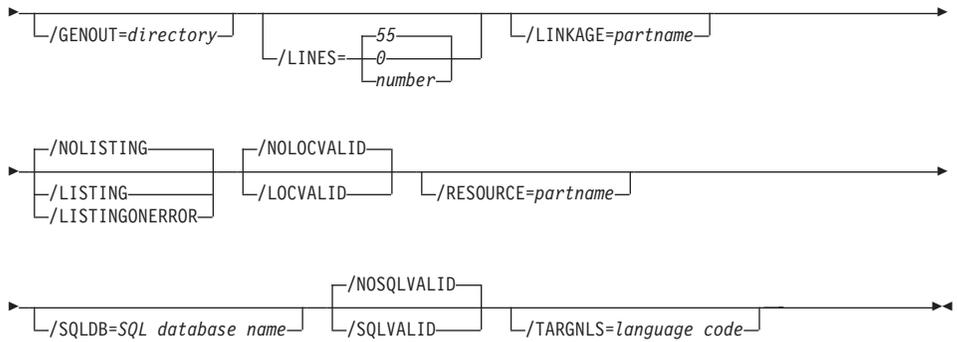
Example

The following is an example of how to use the STOP subcommand:
HPTCMD STOP

VALIDATE subcommand syntax for C++ generation

The VALIDATE subcommand enables you to validate your VisualAge Generator program without actually generating code.





See “Chapter 29. Generation command and option descriptions” on page 297 for the detailed information about the syntax diagram.

Example

Figure 11 illustrates how to use the VALIDATE subcommand for C++ validation.

```
HPTCMD VALIDATE MYGEN /CONFIGMAPNAME="myconfig" /CONFIGMAPVERSION="27.8" /SYSTEM=AIX
```

Figure 11. C++ validation

Part 3. Generating Java programs

Chapter 6. Inputs to Java server program generation

The Java server program generation process uses inputs from several different sources:

- VisualAge Generator part definitions that define the program and its associates.

Refer to the VisualAge Generator Developer online help system for more information about defining parts.

- Generation control files and parts, which are predefined. The generation control files and parts for Java are described in “Part 7. Reference information” on page 207.

These control parts and files contain installation and project-level conventions that specify how the Java server programs and related objects are generated. Generation control parts and files are usually specified at an installation or project level, but they can be specified at any level. You can modify the contents of the generation control parts and files.

If you elect to create a properties file when you generate Java server programs, values from generation options files, linkage tables, and resource association files are built into a properties file that can be accessed by the program at run time. If you do not generate a properties file during generation you can modify the settings in the default properties file, `vgj.properties`, or use it as an example to create your own properties file.

Settings generated into the properties file control NLS defaults, database defaults, linkages, and resource associations.

Note: Currently, Java server programs can only be generated for Windows NT. A properties file is used to control the run-time behavior of the Java program. See “Chapter 30. Java properties files” on page 359 for more information on properties files.

Refer to the *VisualAge Generator Server Guide for Workstation Platforms* document for information about resource association properties and supported file types for the Windows NT environment.

Inputs for Java generation

The following tables list the inputs for Java generation.

Table 4 lists the Java inputs for the supported environments.

Table 4. Inputs for Java generation

Element	File name and extension	Environment
Part definition	Stored in repository	Windows NT
User-defined option files (e.g. generation options file)	Stored in repository	Windows NT
Linkage table	Stored in repository	Windows NT
Conversion tables	ELAxxxxx	Windows NT
Resource association part	Stored in repository	Windows NT

Generation Options parts

When you generate Java server programs, the generation options you select define your generation output. You can set generation options using the Generation Options notebook or you can create a Generation Options part. If you create a Generation Options part, you can specify some options that are not available from the notebook. See “Chapter 24. Generation options parts” on page 209 for information about generation options parts.

Linkage table parts

Linkage table parts are required if the program contains calls to VAGen and non-VAGen programs. Refer to the *VisualAge Generator Client/Server Communications Guide* for information about linkage tables.

Resource associations part

The contents of the resource association file specifies the attributes of accessed resources such as, location, system, file type, data set name, etc. to be used by the program when it accesses those resources.

Properties from the resource associations part, used at runtime by generated Java programs, are stored in a Java properties file during generation if you specify the Properties file (/GENPROPERTIES) option.

See “Chapter 30. Java properties files” on page 359 for a list of related properties. Refer to the *VisualAge Generator Server Guide for Workstation Platforms* for information about resource association settings in Java properties files and supported associate file types for the Windows NT environments.

Conversion tables

Java server programs can access sequential data stored as a record. Conversion tables specified in the properties file are used to convert data when the record's data comes from a message queue. That conversion table may be specified in the MQ file's entry in resource association properties.

Refer to the *VisualAge Generator Client/Server Communications Guide* for information on how to define a custom conversion table for use with code pages associated with other languages for client/server programs.

Chapter 7. Outputs of Java program generation

The outputs of the generation process are Java source and related objects needed to prepare and run your program. These outputs contain the information necessary to transfer files and start the appropriate link processes.

If you are generating a Web transaction program, output includes Java source and related objects. For more information on the generated files, refer to “Chapter 17. Outputs of Web transaction program generation” on page 183.

Unlike Java GUI clients, Java server programs are not generated into the ENVY library. Instead they are generated as .java files and stored in the specified directory. Files that might be outputs of generation are shown in Table 5.

Table 5. Generation outputs for Windows NT.

Element	Method name	Class	File name	Uploaded	Environment
Java source for each program generated		<i>progrname</i>	<i>progrname.java</i> ¹	Yes	Windows NT
Java source for each function generated	<i>funcF</i> ²	<i>progrname</i>	<i>progrname.java</i>	Yes	Windows NT
Java source for flow statements for each function generated	<i>flowF</i> ²	<i>progrname</i>	<i>progrname.java</i>	Yes	Windows NT
Java source for each working storage record		VGWkgW ³	VGWkgW.java	Yes	Windows NT
Java source for each file record		VGFileR ⁴	VGFileR.java	Yes	Windows NT
Java source for each SQL record (JDBC)		VGJdbcJ ⁵	VGJdbcJ.java	No	Windows NT
Java source for each UI record		VGUirU ⁶	VGUirU.java	No	Windows NT
Java source for each MQ record		VGMsgM ⁷	VGMsgM.java	Yes	Windows NT

Table 5. Generation outputs for Windows NT. (continued)

Element	Method name	Class	File name	Uploaded	Environment
Java source for each redefined record		VGRedC ⁸	VGRedC.java	Yes	Windows NT
Java source for each DLI record		VGDliD ⁹	VGDliD.java	Yes	Windows NT
Java source for each table		VGTblT ¹⁰	VGTblT.java, VGDataT.tab	Yes	Windows NT
Preparation file for program progname			<i>prognamejs.cmd</i>	Yes	Windows NT
Build command that places the generated .java files prior to compile			<i>prognamejsz.bat</i>	No	Windows NT
Batch command file that invokes the Java compiler			<i>prognamejsz.bat</i>	Yes	Windows NT
File transfer protocol (FTP) control file			<i>prognamejs.ftp</i>	No	Windows NT
Properties file			<i>progname.properties</i>	Yes	Windows NT

Table 5. Generation outputs for Windows NT. (continued)

Element	Method name	Class	File name	Uploaded	Environment
---------	-------------	-------	-----------	----------	-------------

Notes:

¹*progname* indicates the component of the class name; it is the same as name of the program being generated.

²*F* indicates a component of the method name and is the same as the function name.

³*W* indicates a component of the class name and the file name; it is the same as the working storage record name.

⁴*R* indicates a component of the class name and the file name; it is the same as the file record name.

⁵*J* indicates a component of the class name and the file name; it is the same as the SQL (JDBC) record name.

⁶*U* indicates a component of the class name and the file name; it is the same as the UI record name.

⁷*M* indicates a component of the class name and the file name; it is the same as the MQ record name.

⁸*C* indicates a component of the class name and the file name; it is the same as the redefined record name.

⁹*D* indicates a component of the class name and the file name; it is the same as the DL/I record name.

¹⁰*T* indicates a component of the class name and the file name; it is the same as the table name.

Generated objects are stored in the directory specified on the /GENOUT generation option. If /GENOUT specifies a location that is not valid or if /GENOUT is not specified, the generation objects are stored in the directory where the server process is running.

Chapter 8. Preparation process for Java generation

When generation is completed, the generated outputs must be prepared for run time, just as you prepare programs you might write yourself.

Preparation takes place automatically unless you specify the `/NOPREP` option. If you specify the `/NOPREP` option, preparation takes place as a separate process. The `/PREP` option is the default.

The preparation process includes the following steps:

- Transferring parts to the target environment, if needed
- Unicode conversion, if needed
- Running the compiler

The `PREPARE` subcommand can also be used to restart preparation if the preparation process is not successful in a manner that does not require the parts be generated again. The `PREPARE` subcommand is used for preparing programs generated using any VisualAge Generator Developer.

A `.cmd` file is generated for each program, table, or UI record being generated. A `.cmd` file is run unless the `/NOPREP` option is specified.

Preparing Java programs

The `FCEJBLD.EXE` does the following processing:

1. If the development machine is different from the target machine, the `progrnamejs.ftp` file is used to control transfer of all the required files to the target machine. The `progrnamejsz.cmd` starts preparation for the server program. The `progrnamejsz.bat` copies the necessary files to the package directory. The `progrnamejsz.bat` file invokes the Java compiler. The transfer includes the generated output of any tables, UI records used by the program as well as any generated properties files, which contain settings from generation options, resource associations, linkages stored in the properties file..
2. To start the compile process, the `progrnamejsz.bat` is run on the Windows NT. The compile process compiles each program and UI record.

If the target machine is a remote Windows NT the `progrnamejsz.bat` must be run manually to start the make process.

Refer to the *VisualAge Generator Server Guide for Workstation Platforms* for information on how set up your Windows NT system and how to run your prepared program.

Analyzing preparation errors

When you analyze the preparation errors for the Windows NT environment, do the following:

1. Verify that the *progrnamejsz.bat* file contains the correct compile information for your program.
2. Use the SET command to display information about environment variables settings (e.g. CLASSPATH) for a Windows NT session.

Common preparation errors

The most common preparation problem is that the preparation job will not run on the target system.

Preparation jobs cannot run successfully on the target system when any of the following requirements are not satisfied:

- Because Java is case sensitive, be sure to verify the case for the specified values for the following options:
 - /DESTDIR
 - /DESTHOST
 - /DESTUID
 - /DETPASSWORD
- The user specified on the /DESTUID option must have read and execute authority to the target machine (specified by the /DESTHOST option) and must also have write authority to the target directory (specified by the /DESTDIR option).

The example below shows the command you type at a command prompt where you are running the generator to check for read and execute authority.

```
rexec <desthost> -l <destituid> -p <destopassword> df
```

You can check for write authority by doing the following:

1. Log on to the Windows NT machine using the values you are specifying for the /DESTUID and /DETPASSWORD options.
2. Change to the directory you are specifying on the /DESTDIR option using the cd command.
3. Test your access by creating a file.

The example below shows an example command that lists the files and redirects the list to a file. This is one way to create a file.

```
dir * > temp.file
```

If you are able to create the file, you have write authority.

Note: You can specify relative or full path names for the /DESTDIR option, but fully qualified path names are recommended.

If you use your own preparation process, be sure to name the files correctly. If you use the preparation process provided with VisualAge Generator, it moves the files to the target environment. You must be sure to name the files correctly. If you do not specify the /NOPREP generation option, the VisualAge Generator Developer moves the files for you and correctly names them.

Note: Ensure that the appropriate prerequisites are properly installed. See <http://www.ibm.com/software/ad/vsgen> for additional information.

Chapter 9. Command interface for Java generation

You can issue the VisualAge Generator Developer subcommands from a system prompt or from within a command file. The command HPTCMD implements the command interface. Subcommands are specified with the HPTCMD command and are followed by any required keywords and options. Comments can be imbedded in the commands. Comments begin with the characters `/*` and end with the characters `*/`.

Command processing can be started explicitly by issuing the START subcommand or implicitly by issuing any other VisualAge Generator Developer subcommand. The command continues running until it is ended, either by issuing the STOP subcommand or by closing the Generation Monitor window.

Starting the command opens a Generation Monitor window. The Generation Monitor window displays the command currently being processed and provides information showing what stage of generation the process has reached. You can cancel the currently processing command from the Generation Monitor window. Closing the Generation Monitor window ends any command currently being processed.

If you are generating programs using a generation server, refer to the *VisualAge Generator System Development Guide* for information about starting and stopping the Generation Monitor.

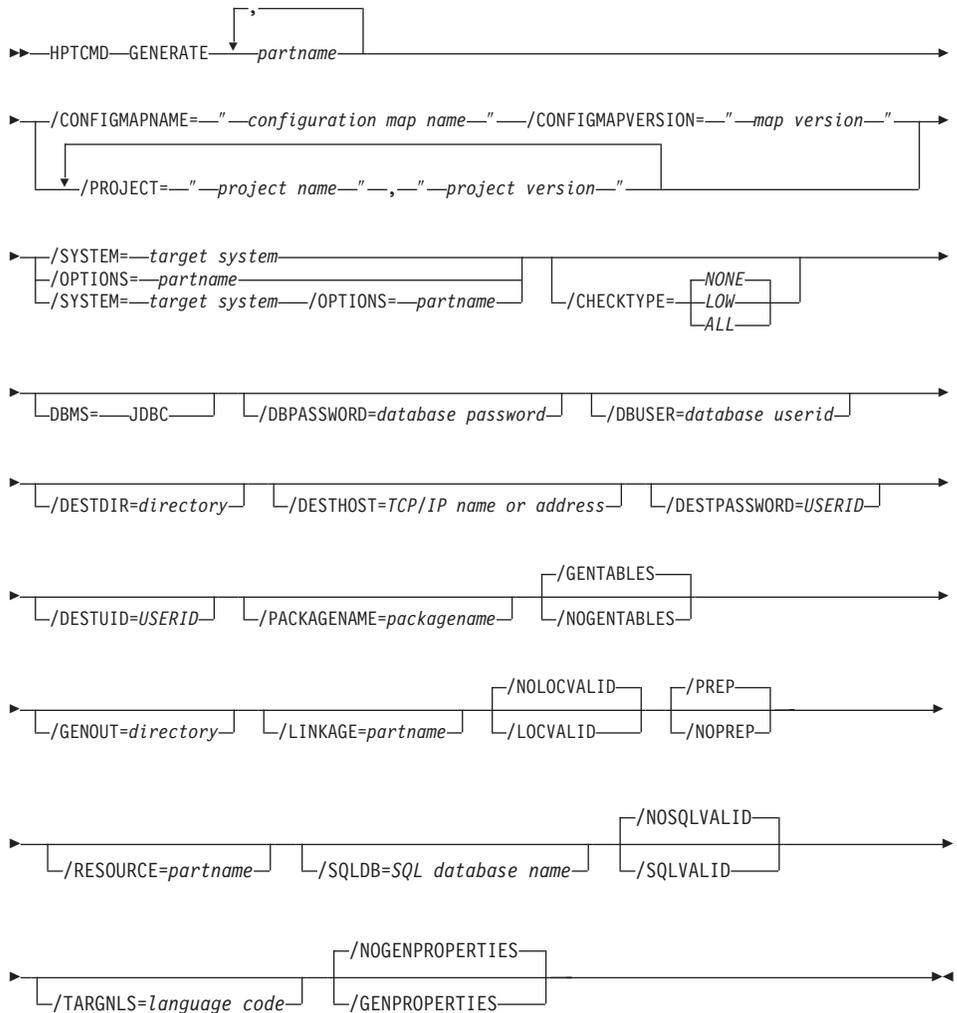
Note: If you are generating a program that uses DBCS, you must run the commands on a machine that is DBCS-enabled.

GENERATE subcommand syntax for Java generation

For Java generation, the GENERATE subcommand enables you to generate a program, table, or UI record. You can also specify options that affect how a part is generated.

The following syntax diagram shows the options required for the GENERATE subcommand.

Note: The `/CONFIGMAPNAME`, `/CONFIGMAPVERSION`, and `/SYSTEM` options require a value when you generate from the command interface using VisualAge Generator on Smalltalk. The `/PROJECT` and `/SYSTEM` options require a value when you generate from the command interface using VisualAge Generator on Java.



See “Chapter 29. Generation command and option descriptions” on page 297 for the detailed information about the syntax diagram.

Example

Figure 12 on page 45 illustrates the GENERATE subcommand for Java program generation; at a command prompt it must be entered in one continuous line. If you are using VisualAge Generator on Java, use /PROJECT instead of /CONFIGMAPNAME and /CONFIGMAPVERSION.

```
HPTCMD GENERATE MYPGM
/CONFIGMAPNAME="myconfig" /CONFIGMAPVERSION="27.8"
/SYSTEM=JAVAWINNT
/COMMENTLEVEL=0
/DESTDIR='/u/pancho'
/DESTHOST=quixote.cary.ibm.com
/DETPASSWORD=myspassword
/DESTUID=pancho
/OPTIONS=MYOPT
/GENOUT=D:\pancho\gen /NUMOVFL
/PACKAGENAME="my.pkg"
```

Figure 12. GENERATE subcommand for Java

Verify the case for the specified values for the following options:

```
/DESTDIR
/DESTHOST
/DESTUID
/DETPASSWORD
```

See “Chapter 24. Generation options parts” on page 209 for information about setting generation options.

See “Chapter 31. Analyzing return codes and errors” on page 367 for information about generation return codes.

GENERATE subcommand examples for Java

The example in this section illustrates how you use the GENERATE subcommand and includes the GENERATE syntax that is used. This example is shown as having several lines so that it will fit page limitations, but the text must be entered as one line when you use the command.

Generating for Windows NT

Figure 13 shows as example that generates a program for use in the Windows NT environment.

```
HPTCMD GENERATE PRGM1 /PROJECT="myproject", "27.8"
/SYSTEM=JAVAWINNT
/LINKAGE=linktbl
/DESTDIR=c:\vggenout
/DESTHOST=nthost1
/DESTUID=sleepy
/DETPASSWORD=secret
/PACKAGENAME="my.pkg"
```

Figure 13. Generating for Windows NT

START subcommand syntax

The START subcommand starts the server process that runs HPTCMD subcommands.

▶—HPTCMD—START—▶

See “Chapter 29. Generation command and option descriptions” on page 297 for the detailed information about the syntax diagram.

If you do not issue the START subcommand and the server process is not already running, any HPTCMD subcommand you issue, except for the STOP subcommand, starts the server process automatically.

Note: If you do not issue the START subcommand and you direct your output to a location other than STDOUT, the server process inherits the STDOUT location. This causes the STDOUT file to be locked by the server until an HPTCMD STOP command is issued. To avoid this, always issue an HPTCMD START command, either at the command line or by placing the command at the beginning of your command file.

Example

The following is an example of how to use the START subcommand:

```
HPTCMD START
```

STOP subcommand syntax

The STOP subcommand stops the server process that runs HPTCMD subcommands.

▶—HPTCMD—STOP—▶

See “Chapter 29. Generation command and option descriptions” on page 297 for the detailed information about the syntax diagram.

The server process continues to run until it receives a STOP subcommand or until the Generation Monitor window is closed. When you specify the STOP subcommand, the server process completes all previously issued subcommands before stopping. If the server process is stopped as a result of the Generation Monitor window closing, the stop occurs immediately, ending any subcommand currently being processed.

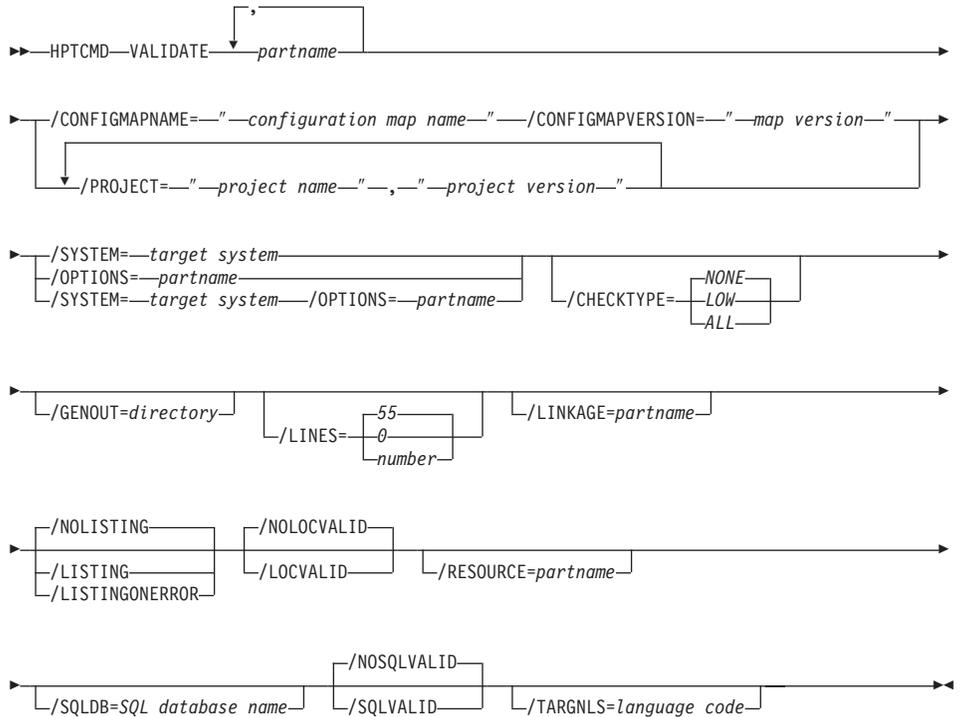
Example

The following is an example of how to use the STOP subcommand:

```
HPTCMD STOP
```

VALIDATE subcommand syntax for Java generation

The VALIDATE subcommand enables you to validate your VisualAge Generator program without actually generating code.



See “Chapter 29. Generation command and option descriptions” on page 297 for the detailed information about the syntax diagram.

Example

Figure 14 on page 48 illustrates how to use the VALIDATE subcommand for Java validation.

```
HPTCMD VALIDATE MYGEN /PACKAGENAME="my.pkg" /CONFIGMAPVERSION="27.8" /SYSTEM=JAVAWINNT
```

Figure 14. Java validation

Part 4. Generating COBOL programs

Chapter 10. Inputs to COBOL generation

The COBOL program generation process uses inputs from several different sources:

- VisualAge Generator part definitions that define the program, tables, processes, statement groups, records, generation options, linkage tables, resource associations, link edit controls, and BIND controls. Refer to the VisualAge Generator Developer online help system for more information about defining parts.
- Generation control files. The generation control files for COBOL include templates, reserved word files, and conversion tables.

Inputs for COBOL generation

Table 6 shows the inputs to COBOL generation, including control parts, files and templates, and the outputs from generation that are used to prepare the generated parts for running on the target system.

Table 6 lists the inputs for the supported COBOL environments.

Table 6. Inputs for COBOL generation

Element	File name and extension	Environment	Notes
Programs, maps, map groups, tables	Stored in repository	All	
Variables	Set in the <i>hpt.ini</i> file	All	Variables that can affect the generation process. Refer to the <i>VisualAge Generator Installation Guide</i> for a description of environment variables.
Generation options parts	Stored in repository	All	
Linkage table parts	Stored in repository	All	

Table 6. Inputs for COBOL generation (continued)

Element	File name and extension	Environment	Notes
Link edit parts	Stored in repository	IMS BMP IMS/VS MVS batch CICS for MVS/ESA MVS/TSO VM batch VM CMS VSE batch CICS for VSE/ESA	
Reserved word file	xxxxxxx.RSV ¹	All	
Conversion tables	ELACNxxx ² .CTB ³ ELACNxxx.DLL ⁴	IMS BMP IMS/VS MVS batch CICS for MVS/ESA MVS/TSO VM batch VM CMS VSE batch CICS for VSE/ESA CICS for OS/2	
Bind control part	Stored in repository	IMS BMP IMS/VS MVS batch CICS for MVS/ESA MVS/TSO	
Resource associations part	Stored in repository	All	
Templates	xxxxxxx.TPL	All	

Notes:

¹xxxxxxx indicates an 8-character file name.

²xxx indicates a 3-character mnemonic for a language, such as *jpn* for Japanese.

³ ELACNxxx.CTB is used when you develop on an OS/2 system.

⁴ ELACNxxx.DLL is used when you develop on a Windows NT system.

Depending upon the environment and use of the COBOL program, certain interim preparation steps occur. For further information on the preparation steps, see “Chapter 14. Preparation process for COBOL generation” on page 153.

Generation options parts

You can optionally specify generation options parts to customize generation of COBOL programs. See “Chapter 24. Generation options parts” on page 209 for information about generation options parts.

Linkage table parts

Linkage table parts are required if a program contains external calls or you want to change the default “call linkage” type. A linkage table specifies the following:

- The linkage conventions to be used for calling a program
- Whether a CICS CREATX service call starts a local or remote CICS transaction
- The linkage conventions to be used for implementing a DXFR transfer between host programs
- Whether a CICS file is to be accessed as a local or remote file

External calls are calls to non-VisualAge Generator programs or calls to VisualAge Generator server programs on remote systems. Refer to the *VisualAge Generator Client/Server Communications Guide* for information about linkage tables.

Link edit parts

Link edit parts are used for the MVS, VSE, and VM environments. Preparation templates shipped with VisualAge Generator contain default linkage editor control statements for generated programs. You must also define additional linkage editor control statements in a link edit part if a program calls or is called by other programs using static COBOL calls.

For further information, see “Chapter 26. Link edit parts” on page 245.

Reserved-word file

The reserved word file contains names that are considered reserved by the generation process. If a name used in a program matches an entry in the reserved word file, the name is assigned an alias in the generated source. However, an alias cannot be used for the following:

- Programs
- Tables
- Map groups
- File names

Reserved words can be used for a file name in the following cases:

- When the target system is a CICS environment
- When the target system is not a CICS environment and the /FILETYPE option is not one of the following:
 - OS2COBOL
 - SEQ
 - VSAM

At installation time, the reserved word file contains reserved words for COBOL, SQL, CICS, and VisualAge Generator. Your system administrator can add new reserved words to the file if required. Most installations do not need to modify this file.

The file contains two types of records:

Comment statements

A statement with an asterisk (*) in column one. You can add additional comment statements to the file.

Reserved-word statement

A statement with a word starting in column one. The word extends to the first blank. You can add additional reserved word statements to the file.

If a program, map group, table, or file name matches a reserved word, generation ends with an error. Any other part name that matches a reserved word is assigned an alias in the generated object.

In addition to specific reserved words, the default file contains the following generic reserved words:

- DFH*
- EIB*
- SQL*

When the reserved word ends in an asterisk, any name whose initial characters match the string preceding the asterisk is assigned an alias. These generic reserved words are used to avoid conflict between program variable names and program variable names assigned by the CICS translator and DB2, DB2/2, DB2/VSE, or SQL/DS VM precompilers.

The default reserved word file is EFK2RSV.RSV. You can specify a different file name using the /RESVWORD generation option. See "Chapter 29. Generation command and option descriptions" on page 297 for more information on specifying a reserved word file name using the /RESVWORD generation option.

The file name must be specified in one of the following ways:

- A fully qualified OS/2 or Windows NT file name. Symbolic parameters are permitted in the directory specification. See “Chapter 12. Symbolic parameters” on page 123 for more information about symbolic parameters.
- The name of a file located in a directory specified in the DPATH environment variable of the server process. See “Chapter 15. Command interface for COBOL generation” on page 167 for more information about the server process.

If you want to modify the reserved word file, make a copy of the file and edit the copy. After the copied file is modified, use the /RESVWORD generation option to point to the modified file.

Conversion tables

Conversion tables are used to convert ASCII data text to EBCDIC data in generated objects when objects are transferred from the workstation to MVS, OS/400, VM, or VSE systems for preparation. They are also used to convert OS/2 ASCII to the corresponding Windows or UNIX ASCII or to convert Windows ASCII to the corresponding OS/2 or UNIX ASCII. Conversion tables are specified using the /CONTABLE generation option. See “Chapter 29. Generation command and option descriptions” on page 297 for more information about the /CONTABLE generation option and about conversion tables.

Refer to the *VisualAge Generator Client/Server Communications Guide* for information on how to define a custom conversion table for use with code pages associated with other languages for client/server programs.

BIND control parts

BIND control parts are only required for DB2 programs generated for MVS systems.

Whenever a DB2 program is prepared for run time, there are two steps included in the preparation JCL or preparation REXX procedure:

DB2 precompiler step

Creates a file called a database request module (DBRM) for the program

BIND step

Binds the DBRM with the DBRMs of other programs into DB2 plans.

The DBRMs for all programs or non-VisualAge Generator programs that run together as one transaction or one batch job must be bound into a single DB2 plan.

For further information on the BIND control part, see “Chapter 27. BIND control parts” on page 257.

Resource associations part

See “Chapter 28. Resource associations part” on page 265 for information on the resource associations part.

Chapter 11. Templates for COBOL generation

Templates are needed to generate files used to prepare parts for run time and to run a program. All of the template files required for generation are shipped with VisualAge Generator Developer. See “Types of templates” on page 58 for the templates provided with VisualAge Generator.

The file extension for template files is TPL. During installation, these sample template files are placed in the directory C:\Program Files\Vast\bin for the Smalltalk version of VisualAge Generator or C:\IBMJava\IDE\program for the Java version. The installation procedure updates the OS/2 environment variable DPATH or the Windows NT environment variable PATH to include this directory.

If the /TEMPLATES generation option is specified, only the specified directory is searched for templates. The specified directory must contain copies of all the templates. If the /TEMPLATES generation option is not specified, the directories listed by the DPATH or the PATH environment variable of the process currently running the HPTCMD command are the directories that are searched for templates.

The templates contain symbolic parameters, which are indicated by the % symbols. Values can be dynamically defined and substituted in place of these symbolic parameters. The substitution values are derived from any of the following sources:

- Definitions of the parts being generated
- Generation options
- /SYMPARM values

See “Chapter 12. Symbolic parameters” on page 123 for more information about symbolic parameters.

In addition to symbolic parameters, some program-related templates contain control statements that indicate the points at which information, like file definitions or program commands, is to be inserted in the output file. The inserted information can also be generated from templates (file definitions) or provided by the program user (BIND commands or link-edit statements).

The control statement indicating an insertion point has the format ?KEYWORD?, where the keyword indicates the type of information to be produced at that point. The following control statement keywords are supported:

?DD? Insert MVS DD statements

?ALLOC?

Depending on the environment, whether MVS/TSO, VM CMS, or VM batch, appropriate allocation statements are inserted

?DLBL?

Insert VSE DLBL statements

?LINK?

Insert link-edit control statements

Types of templates

Templates can be modified to meet installation requirements for preparing and running programs. A system administrator or project leader usually does these modifications. See “Modifying templates” on page 82 for all the detailed information about modifying templates and for examples of how the templates are processed during generation.

See the following sections for detailed information about the different types of templates provided with the VisualAge Generator Developer.

Preparation templates

If you specify the /PREPFILE generation option in VisualAge Generator, a preparation script is created. The name of the script is *partname*.PRP. The command file EFKPREP.CMD will always be called for preparation. This command file calls the program that interprets the .PRP control file, then uploads the necessary files and submits the preparation procedure.

See “Chapter 14. Preparation process for COBOL generation” on page 153 for more information about the preparation process.

Preparation script templates

When you generate COBOL programs, the preparation script is created which contains control keywords to transfer the generated source files and preparation files to the target system.

The following preparation files are created for each target system:

Table 7. Preparation executable for each target system

Target system	Preparation executable
MVS	JCL
VSE	JCL
VM	REXX
OS/2 CICS	REXX
OS/400	CL

The profile templates listed in Table 8 contain information on the following:

- Environment-specific information for the transfer
- How each part type is to be moved to the preparation system

The template EFK2OPCA.TPL is the preparation script template for all the environments. The only tag that is in the template is the :CONTROL tag. This template is repeated in the preparation script for each part generated.

The templates EFK2OPP_x.TPL are divided into three sections. Each section is delimited by a tag. The tags are :PROFILE, :TYPE, and :MESSAGES. Each section contains keywords, which must remain in their respective sections or a preparation syntax error will occur.

Table 8. Preparation template description

Section	Keyword	Default	EFK2OPP4.TPL (AS/400)	EFK2OPPC.TPL (VM)	EFK2OPP.V.TPL (VSE)	EFK2OPPM.TPL (MVS)	EFK2OPPO.TPL (OS/2 CICS)	EFK2OPCA.TPL (all)
:PROFILE	TRANSFER_TYPE	%ezetransfertype%	X	X	X	X	X	
	GENOUT	%ezezenout%	X	X	X	X	X	
	SYSTEM	%ezeenv%	X	X	X	X	X	
	PREP_SYSTEM	%mvs; for %mvs;, OS2 for OS/2, OS400 for OS/400, VM for VM, VSE for VSE	X	X	X	X	X	
	CONVERSION_TABLE	%ezeconversiontable,ELAO2437% for OS/2, %ezeconversiontable,ELACNENU% for all others	X	X	X	X	X	
	ENTRY_POINT	1	X	X	X	X	X	
	DELETE_FILES	N	X	X	X	X	X	X
	PREP	D	X	X	X	X	X	X
	SESSION	%ezeprepession%	X	X				
	SENDFILE_OPTIONS	null string (")	X	X	X	X	X	
	QUIET	N	X	X	X	X	X	X
	DESPASSWORD	%ezeprepdespassword%	X	X		X	X	

Table 8. Preparation template description (continued)

Section	Keyword	Default	EFK2OPP4.TPL (AS/400)	EFK2OPPC.TPL (VM)	EFK2OPPV.TPL (VSE)	EFK2OPPM.TPL (MVS)	EFK2OPPO.TPL (OS/2 CICS)	EFK2OPCA.TPL (all)
	DESTACCOUNT	%ezeprepredestaccount%	X	X		X	X	
	DESTUID	%ezeprepredestuid%	X	X		X	X	
	DESTHOST	%ezeprepredesthost%	X	X		X	X	
	OS2_DESTDIR	%ezeprepredestdir%					X	
	OS400_DESTLIB	%ezedestlib%	X					
	VM_FMODE	%vmfmode, A%		X				
	VM_DISK_ADDR	%vmdiskaddr, 191%		X				
	MVS_ALLOC_OPTIONS					X		
	PROJECTID	%ezepid%				X		
	VSE_LIB	%ezevselib%			X			
:TYPE	PART_TYPE		X	X	X	X	X	
	WORKSTATION_EXT		X	X	X	X	X	
	WORKSTATION_EXT_ALT						X	
	IS_BINARY		X	X	X	X	X	
	PREP_SUBMIT		X	X	X	X	X	
	SENDFILE_OPTIONS	Null string (")	X	X	X	X	X	
	CNV_RECORD_LENGTH	80	X	X	X	X	X	
	USE_CSO_CONVERSION	Y	X	X	X	X	X	
	OS400_FILENAME		X					
	OS400_FILENAME_SUFFIX		X					
	MVS_EXT					X		
	MVS_ALLOC_OPTIONS					X		
	VM_FTYPE			X				
	VM_FTYPE_SQL			X				
	VM_RECFM			X				

Table 8. Preparation template description (continued)

Section	Keyword	Default	EFK2OPP4.TPL (AS/400)	EFK2OPPC.TPL (VM)	EFK2OPPV.TPL (VSE)	EFK2OPPM.TPL (MVS)	EFK2OPPO.TPL (OS/2 CICS)	EFK2OPCA.TPL (all)
	VM_LRECL			X				
	VSE_FTYPE				X			
:MESSAGES	MSGCGxxx		X	X	X	X	X	
:CONTROL	HAS_DBCS	%ezedbcse ^v , N%						X
	NAME	%ezembr%						X
	TYPE	%ezeptype%						X
	HAS_SQL	%ezesql, N%						X
	USE_EXT_ALT							X

Preparation Rexx for CICS for OS/2

If you specify the /PREPFILE option and your target system is CICS for OS/2, a preparation REXX file (*partname.RXP*) is created. The preparation REXX file contains the preparation commands for preparing the parts you have requested.

VisualAge Generator provides templates that are partial REXX files. These templates are used during generation to produce a REXX file that performs the preparation steps required for a program, table, or map group. The VisualAge Generator /PREPFILE function uses the following templates to produce the REXX preparation REXX file:

EFK2OPXP

This preparation prologue template for CICS for OS/2 preparation contains the following information:

- Function of the /PREPFILE template
- How to change the preparation prologue template

EFK2OPXC

This preparation control template for CICS for OS/2 preparation

contains a REXX array that is initialized with information about the part to be prepared. It is repeated in the preparation REXX file for each part generated.

EFK2OPXF

This preparation function template contains the REXX function code that uses the preparation control template for preparing a part for execution.

You can change the preparation process by changing the information in the EFK2OPXP or EFK2OPXF templates. However, changes to the EFK2OPXP template are not recommended. See “Procedure name and symbolic parameters” on page 86 for information about required preparation options that should not be changed.

See “Chapter 14. Preparation process for COBOL generation” on page 153 for more information about preparation.

Preparation JCL for MVS or VSE

A preparation JCL file (mbrname.JCP) is created if you specify the /PREPFILE option, and your target system is MVS or VSE.

For VSE generation, the preparation JCL file also contains end-of-job information.

The templates used for creating the preparation JCL file are selected according to the following criteria:

- Type of part being generated
- Target run-time environment
- Types of databases used in the generated program
- Existence of program-specific BIND (prgmname) part for MVS or link-edit (prgmname) part for VSE

Note: For MVS environments, the templates for BIND commands are only used if you have not created a program-specific BIND control part.

For VSE environments, the templates for link-edit statements are only used if you have not created a program-specific link-edit part.

Program templates

Table 9 on page 63 shows the program templates and procedures used for CICS for MVS/ESA, MVS/TSO, and MVS batch preparation. The templates and procedures are selected based on the following information:

- The environment
- Use of DB2 in the program
- Use of DL/I in the program

Table 9. Program templates used for CICS for MVS/ESA, MVS/TSO, and MVS batch preparation

Environment	DB2	DL/I	Template	Procedure	DB2 BIND template
CICS for MVS/ESA	Yes	*	EFK2MPCA	ELAPTCLB	EFK2MBDA
CICS for MVS/ESA	No	*	EFK2MPCB	ELATCL	
MVS/TSO	Yes	No	EFK2MPTB	ELAPCLB	EFK2MBDD
MVS/TSO	No	*	EFK2MPTA	ELACL	
MVS batch	Yes	Yes	EFK2MPBB	ELAPCLB	EFK2MBDA
MVS batch	Yes	No	EFK2MPBC	ELAPCLB	EFK2MBDD
MVS batch	No	*	EFK2MPBA	ELACL	

*No entry in the DL/I column indicates that DL/I use does not affect the selection of this template.

See “Template for statically linked programs” for more information.

Template for statically linked programs

Table 10 shows the template and the procedure used with statically linked programs for the MVS environment.

Table 10. Templates used with statically linked programs on MVS environments

Template	JCL Procedure Called	Function
EFK2MPRE	ELARLINK	Link-edit MVS programs that use static COBOL calls

See “Link edit parts” on page 53 for more information about Linkage Editor control statements.

More about DL/I usage in MVS/TSO and MVS batch

In the MVS/TSO or MVS batch environments, DL/I use is indicated only if the program includes at least one of the following attributes or functions:

- Database PCB, other than for the ELAWORK or ELAMSG synonyms, included in the PSB
- EZEDLPCB or EZEDLPSB special function word passed on a CALL statement
- EZEDLPCB or EZEDLPSB special function word received as a parameter for a called program
- CSPTDLI call

- AUDIT call with PSB specified for program
- EZEDLPCB reference
- GSAM file association (MVS batch only)

Program templates for IMS/VS and IMS BMP preparation

Table 11 shows the program templates and procedures used for IMS preparation. The templates and procedures are selected based on the following information:

- The environment
- Use of DB2 in the program
- Specification of a DB2 work database

Note: DL/I is always used in the IMS/VS and IMS BMP environments.

Table 11. Program templates used for IMS preparation

Environment	DB2	DB2 Work Database	Template	Procedure	DB2 BIND template
IMS/VS	Yes	Yes	EFK2MPIE	ELAPCLB	EFK2MBDB
IMS/VS	Yes	No	EFK2MPIE	ELAPCLB	EFK2MBDA
IMS/VS	No	Yes	EFK2MPID	ELACLB	EFK2MBDC
IMS/VS	No	No	EFK2MPIC	ELACL	
IMS BMP	Yes		EFK2MPIB	ELAPCLB	EFK2MBDA
IMS BMP	No		EFK2MPIA	ELACL	

Program templates for CICS for VSE/ESA and VSE batch preparation

Table 12 shows the program templates and procedures used for VSE preparation. The templates and procedures are selected based on the following information:

- The environment
- Use of SQL in the program
- Use of DL/I in the program

Table 12. Program templates used for VSE preparation

Environment	SQL	DL/I	Template	Procedure	Link-edit template
CICS for VSE/ESA	Yes	Yes	EFK2VPCB	ELACUSTP ELASQLP	EFK2VLCD
CICS for VSE/ESA	Yes	No	EFK2VPCB	ELACUSTP ELASQLP	EFK2VLCC
CICS for VSE/ESA	No	Yes	EFK2VPCA	ELACUSTP	EFK2VLCCB

Table 12. Program templates used for VSE preparation (continued)

Environment	SQL	DL/I	Template	Procedure	Link-edit template
CICS for VSE/ESA	No	No	EFK2VPCA	ELACUSTP	EFK2VLCA
VSE batch	Yes	Yes	EFK2VPBC	ELACUSTP ELASQLP ELADLIP	EFK2VLBD
VSE batch	Yes	No	EFK2VPBD	ELACUSTP ELASQLP	EFK2VLBC
VSE batch	No	Yes	EFK2VPBB	ELACUSTP ELADLIP	EFK2VLBB
VSE batch	No	No	EFK2VPBA	ELACUSTP	EFK2VLBA

Map group templates

Table 13 shows the map group templates and procedures used for MVS preparation. The templates and procedures are selected based on the following information:

- The environment
- The type of object being generated
- Generation options affecting generation of that object type

Table 13. Map group templates used for MVS preparation

Environment	Generated object type	Generation option	Template	Procedure
CICS for MVS/ESA	Online print services COBOL program		EFK2MMCA	ELACL
CICS for MVS/ESA	Map group format module		EFK2MMTF	ELAL
MVS/TSO	Online print services COBOL program		EFK2MMCA	ELACL
MVS/TSO	Map group format module		EFK2MMTF	ELAL
MVS batch	Batch print services COBOL program	/MSP= (SEQ GSAM)	EFK2MMCA	ELACL

Map group templates for IMS/VS and IMS BMP preparation

Table 14 shows the map group templates and procedures used for IMS preparation. The templates and procedures are selected based on the following information:

- The environment
- The type of object being generated
- Generation options affecting generation of that object type

Table 14. Map group templates used for IMS preparation

Environment	Generated object type	Generation option	Template	Procedure
IMS/VS	MFS print services COBOL program		EFK2MMCB	ELACL
IMS/VS	MFS source	/MFSTEST	EFK2MMST	MFSTEST
IMS/VS	MFS source	/NOMFSTEST	EFK2MMSU	MFSUTL
IMS BMP	MFS print services COBOL program	/MSP= (MFS ALL)	EFK2MMCB	ELACL
IMS BMP	MFS source	/MFSTEST	EFK2MMST	MFSTEST
IMS BMP	MFS source	/NOMFSTEST	EFK2MMSU	MFSUTL
IMS BMP	Batch print services COBOL program	/MSP= (SEQ GSAM ALL)	EFK2MMCA	ELACL

Map group templates for CICS for VSE/ESA and VSE batch preparation

Table 15 shows the map group templates and procedures used for VSE preparation. The templates and procedures are selected based on the following information:

- The environment
- The type of object being generated
- Generation options affecting generation of that object type

Table 15. Map group templates used for VSE preparation

Environment	Generated object type	Generation option	Template	Procedure
CICS for VSE/ESA	Online print services COBOL program		EFK2VTCL	ELACUSTP
CICS for VSE/ESA	Map group format module program		EFK2VMFM	ELACUSTP

Table 15. Map group templates used for VSE preparation (continued)

Environment	Generated object type	Generation option	Template	Procedure
VSE batch	Batch print services COBOL program	/MSP=SEQ	EFK2VTCL	ELACUSTP

Note: The link-edit information for map groups generated for VSE is imbedded directly in the template. There is not a separate link-edit template for VSE map group generation.

Table templates

Table 16 shows the map group templates and procedures used for table preparation. The templates and procedures are selected based on the following information:

- The environment
- The type of object being generated

Table 16. Table templates used for preparation

Environment	Generated object type	Template	Procedure
All MVS environments	Table program	EFK2MMCA	ELACL
All VSE environments	Table program	EFK2VTCL	ELACUSTP

Note: The link-edit information for tables generated for VSE is imbedded directly in the template. There is not a separate link-edit template for VSE table generation.

VisualAge Generator tables generated as COBOL programs are system dependent but not environment dependent. A table generated and prepared for one environment can be used in another environment on the same system without generating the table again.

VSE end-of-job template

The VSE end-of-job template is named EFK2VPEJ.TPL. This template is appended to the preparation JCL stream at the end of the generation process. This template indicates to the VSE/POWER queue that generation is complete, and provides a place for the VSE/POWER end-of-job statement.

Preparation CL for OS/400

The VisualAge Generator Developer default templates produce OS/400 control language (CL) programs to manage the task of compiling and binding ILE COBOL programs. The CL programs have an extension of CLP and CLJ in the generation output directory. When the CL programs are transferred to

OS/400, they are placed in the files QVGNCLS and QVGNJOB in the library named on the /DESTLIB generation option. Each generated member for QVGNCLS or QVGNJOB has the same name as the program part from which it was generated. The submitted batch job (.CLJ) compiles and calls the .CLP file that prepares the generated programs. When the CL program is run, generated VisualAge Generator programs are prepared for run time.

All of the templates have the file-name extension .TPL and are originally installed in the directory C:\Program Files\Vast\bin for the Smalltalk version of VisualAge Generator or C:\IBMJava\IDE\program for the Java version. The following are the templates for the OS/400 environment and their functions:

EFK24PBC

Establishes CL program entry and global error monitoring during preparation of one VisualAge Generator part type

EFK24PCL

Provides CL to compile and bind ILE COBOL/400 when the VisualAge Generator part is a CALLED (non-SQL) program type

EFK24PEC

Provides an epilogue to the CL preparation program, such as general purpose error handling, during preparation of one VisualAge Generator part type

EFK24PMN

Provides CL to compile and bind ILE COBOL/400 when the VisualAge Generator part is a MAIN (non-SQL) program type

EFK24PPM

Provides CL to compile and bind ILE COBOL/400 when the VisualAge Generator part is a map group containing printer maps (mapping services program)

EFK24PSC

Provides CL to DB2/400 pre-compile, COBOL compile, and bind ILE COBOL/400 when the VisualAge Generator part is a CALLED program type containing SQL statements

EFK24PSM

Provides CL to DB2/400 pre-compile, COBOL, compile, and ILE COBOL/400 when the VisualAge Generator part is a MAIN program type containing SQL statements

EFK24WCL

Provides CL to compile and bind ILE COBOL/400 when the VisualAge Generator part is a Web transaction (non-SQL) program type

EFK24WSC

Provides CL to DB2/400 pre-compile, COBOL compile, and bind ILE COBOL/400 when the VisualAge Generator part is a Web transaction program type containing SQL statements

The message file CL used for the OS/400 environment has the file-name extension .MSG. CL for the message file is similar to the preparation for VisualAge Generator program and map group print map part types. The message file preparation CL that is produced from templates prepares a VisualAge Generator message table part when started. During the preparation phase, message table parts are converted to OS/400 message file objects to be used by VisualAge Generator programs. CL message templates are used for the OS/400 environment as follows:

EFK24TCM

Provides CL to create the message file

EFK24TAM

Provides CL to add a message to the message file

EFK24TEM

Provides an epilogue to the CL preparation program, such as general purpose error handling, during preparation of one VisualAge Generator part type

The run-time file CL, used for client/server programs in the OS/400 environment, has the file extension .CLR. During the preparation phase, the run-time CL source is copied to the file QVGNCLS in the destination library, with the name of prgname_R. It is then compiled.

EFK24EBC

Provides the CL to add libraries to the client/server job and to start commitment control if this is the first server program called by a client. Customize this template to add any libraries that the server program needs for data or programs.

EFK24EEC

Provides an epilogue to the run-time CL to handle errors that occur.

Batch jobs used for the OS/400 environment have the file extension .CLJ. OS/400 batch jobs are used to create and then run the preparation CL programs previously defined by the CL templates. The batch job stream is produced from the following templates:

EFK24PBJ

Establishes the database batch job entry. This template must contain the //BCHJOB command. This template is similar to the MVS JOB card template. This template can also contain job setup commands, such as library list alterations and job value changes.

Note: The generation option `/JOB CARD=` value can be used to override this template name with one of your own choosing, enabling easy customization to batch job parameters that control the job definition.

EFK24TMJ

Provides CL commands to create the message CL preparation program, and then runs it. This template is used when the VisualAge Generator part is a message file.

EFK24PEJ

Provides CL commands to create the program CL preparation program, and then runs it. This template is used when the VisualAge Generator part is a program or a print map.

See “Chapter 14. Preparation process for COBOL generation” on page 153 for more information about the preparation process.

Preparation REXX for VM

If you specify the `/PREPFILE` option and your target system is VM CMS or VM batch, a preparation REXX exec file (*partname.RXP*) is created. The preparation REXX exec contains the preparation commands for preparing the parts you have requested.

The parts can be any of the following:

- Programs
- Map groups
- Tables

VisualAge Generator provides templates that are partial REXX files. These templates are used during generation to produce a REXX file that performs the preparation steps required for a program, table, or map group. The VisualAge Generator `/PREPFILE` function uses the following templates to produce the VM REXX preparation REXX file:

EFK2CPXP

This preparation prologue template for VM preparation contains the following information:

- Function of the `/PREPFILE` template
- How to change the preparation prologue template

EFK2CPXC

This preparation control template for VM preparation contains a REXX array that is initialized with information about the part to be prepared. It is repeated in the preparation REXX exec for each part generated.

EFK2CPXF

This preparation function template contains the REXX function code that uses the preparation control template for preparing a part for execution.

You can change the preparation process by changing the information in the EFK2CPXP or EFK2CPXF templates. However, changes to the EFK2CPXC template are not recommended. See “Procedure name and symbolic parameters” on page 86 for information about required preparation options that should not be changed.

See “Chapter 14. Preparation process for COBOL generation” on page 153 for more information about the preparation process.

BIND command templates

BIND commands are required only for DB2 programs.

BIND command templates are used to build a default BIND command file (*prgmname.BND*) for a program if a program-specific bind control part does not exist.

Link-edit templates

Link-edit templates are required for VSE and VM programs.

VSE programs have separate templates containing link-edit information. For VSE map groups and tables, the link-edit information is included in the regular templates.

MVS programs do not have separate link-edit templates. For MVS programs, the link-edit information is included in the cataloged procedures shipped with VisualAge Generator Server for MVS, VSE, and VM. VM programs, map groups and tables have separate templates containing link-edit information. See “Link-edit templates for VM” for more information.

See “Program templates for CICS for VSE/ESA and VSE batch preparation” on page 64 for more information about which link-edit template is used.

Link-edit templates for VM

The link-edit templates used to create default linkage editor control statements when generating for VM target environments are listed in Table 17.

Table 17. Linkage Editor control statement templates for VM programs

Environment	SQL	Template
VM CMS	Yes	EFK2CLKD
VM batch		

Table 17. Linkage Editor control statement templates for VM programs (continued)

Environment	SQL	Template
VM CMS	No	EFK2CLKC
VM batch		

The name of the linkage editor control statement file templates used to generate the default linkage editor control statements for map groups and tables are listed in Table 18.

Table 18. Linkage Editor control statement templates for tables and map groups

Object type	Template
Map group format module	EFK2CLKA
Map group object module	EFK2CLKB
Table object module	EFK2CLKB

CICS table templates

CICS table templates enable automatic updating of the program and transaction control table entries that define programs to CICS for the CICS for MVS/ESA, CICS for VSE/ESA, and CICS for OS/2 environments.

See “Templates for CICS table definition” on page 73 for information on the templates used for CICS table generation.

Automatic update of program and transaction tables is not available for the CICS for OS/2 environment. The CICS definitions must be updated before using the generated programs. See “CICS for OS/2 table entries” on page 163 for information on modifications needed for the CICS for OS/2 table entries. CICS tables are generated for the CICS for OS/2 environment when the /CICSENTRIES generation option is specified with a value other than NONE. The same format is used for CICS for OS/2 table generation whether you specify RDO or MACRO for the /CICSENTRIES generation option.

For CICS for MVS/ESA and CICS for VSE/ESA environments, the table entries are transferred to the host by the preparation process. The CICS tables on the host are not automatically updated with the generated definitions. The generated definitions might need to be modified before being used. Substitute the actual transaction name in place of the symbol in the template. You are responsible for adding the definitions to the appropriate CICS tables on the host systems where the program is run.

Different templates support generation of either macro or resource definition online (RDO) definitions for CICS for MVS/ESA and CICS for VSE/ESA. The

value specified for the /CICSENTRIES generation option lets you select the format used for CICS table generation. The following values are valid:

MACRO

Generates a program properties table (PPT) entry for each COBOL program and FM module. In addition, a program control table (PCT) entry is generated for each program with type main transaction or main batch, and for the transaction ID specified as the segmented transaction ID specified at generation. The table entries are generated from templates.

The MACRO option can be specified for any CICS for MVS/ESA or CICS for VSE/ESA environment. It is recommended that you use RDO whenever possible.

RDO Generates resource definition online (RDO) definitions containing program properties information for each COBOL program and FM module. In addition, the definition for programs with type main transaction or main batch also contains program control and group information for the transaction associated with the program.

The RDO option can be used with CICS for VSE/ESA systems and with CICS/ESA systems on MVS. CICS for MVS/ESA V2 systems do not support using a batch job to add RDO table entries. For CICS for MVS/ESA V2, RDO table entries must be added online.

It is recommended that you use RDO whenever possible.

NONE

Does not generate CICS program and transaction definitions.

The generated table entries are written to files with the name of the part as specified on the GENERATE subcommand, and the following extensions:

- PPT
- PCT

Templates for CICS table definition

Table 19 shows the templates used for CICS table generation:

Table 19. Templates used for CICS table generation

Object type	System	Format	Program template	Transaction template
COBOL	MVS	MACRO	EFK2MCPA	EFK2MCCM
Map group format module	MVS	MACRO	EFK2MCPB	
COBOL	MVS	RDO	EFK2MCPC	EFK2MCCR

Table 19. Templates used for CICS table generation (continued)

Object type	System	Format	Program template	Transaction template
Map group format module	MVS	RDO	EFK2MCPD	
COBOL	VSE	MACRO	EFK2VCPA	EFK2VCCM
Map group format module	VSE	MACRO	EFK2VCPB	
COBOL	VSE	RDO	EFK2VCPC	EFK2VCCR
Map group format module	VSE	RDO	EFK2VCPD	
COBOL	OS/2	CICS for OS/2	EFK2OCPA	EFK2OCCT
COBOL	OS/2	CICS for OS/2	EFK2OCPB	

Note: Transaction templates are used only when generating main transaction or main batch programs.

JOB statements

The JOB statement file contains the JOB statement used for jobs that prepare programs for run-time and for sample run-time JCL. Like other templates, the JOB statement file can contain symbolic parameters that can be substituted with values specified using the /SYMPARM generation option. However, JOB statements differ from other template files in the way you specify use of a non-default file. To use a JOB statement file other than one of the supplied default files, you specify the JOB statement file name using the /JOBCARD generation option.

Your system administrator or project leader can tailor the JOB statements to implement local naming conventions or program preparation processes.

The system administrator or project leader can also insert new symbolic parameters into the JOB statements. If new symbols are added, you can provide the values for these symbols by specifying the values using the /SYMPARM generation option.

For MVS environments, the default JOB statement file name is EFK2MJOB.TPL; for VSE environments, EFK2VJOB.TPL.

Run-time file templates

If you specify the /RUNFILE generation option for program generation, and your target system is MVS batch, IMS BMP, or VSE batch, the generation function creates sample JCL for running the program. The file name is *prgname.JCX*.

If you specify the /RUNFILE generation option for program generation, and your target system is MVS/TSO, the generation function creates a sample CLIST file. The file name is *prgmname.CLX*.

If you specify the /RUNFILE generation option for program generation, and your target system is VM, the generation function creates a sample REXX exec for running the program. The file name is *prgmname.RXX*.

The templates used for creating these sample run-time files are selected according to the following criteria:

- The environment
- Whether the program is a main or called program
- Use of DB2, DB2/VSE, or SQL/DS VM in the program
- Use of DL/I in the program

Note: The /RUNFILE generation option is valid only for the following environments:

- MVS/TSO
- MVS batch
- IMS BMP
- VM CMS
- VM batch
- VSE batch

JCL templates

Table 20 shows the JCL templates and procedures used for running MVS batch programs. The templates and procedures are based on the following information:

- The environment
- Whether the program is a main or called program
- Use of SQL in the program
- Use of DL/I in the program

Table 20. Templates used for running MVS batch programs

Environment	Program	SQL	DL/I	Template	Procedure called by template
MVS batch	Main	Yes	Yes	EFK2MEBB	DLIBATCH
MVS batch	Main	Yes	No	EFK2MEBD	
MVS batch	Main	No	Yes	EFK2MEBC	DLIBATCH
MVS batch	Main	No	No	EFK2MEBE	

Table 20. Templates used for running MVS batch programs (continued)

Environment	Program	SQL	DL/I	Template	Procedure called by template
MVS batch	Called	*	*	EFK2MEBA	

*The use of the template is not affected by either SQL or DL/I usage.

Each of the JCL templates contains a ?DD? line that starts in column one. This line indicates where the generation function is to insert the DD statements for files and databases used by the program. Move this line within the JCL template to control where the DD statements are placed. If you remove the line that contains ?DD?, the DD statements are not included in the generated JCL.

EFK2MEBA

For called programs, only DD statements are generated. These statements can be incorporated into the sample run-time JCL of any main program that calls the program.

More about DL/I use for MVS batch: In the MVS batch environment, DL/I usage is indicated if the program includes at least one of the following attributes or functions:

- Database PCB, other than for the ELAWORK or ELAMSG synonyms, included in the PSB
- EZEDLPCB special function word passed on a CALL statement
- EZEDLPSB special function word passed on a CALL statement
- EZEDLPCB or EZEDLPSB special function word received as a parameter for a called program
- CSPTDLI call
- AUDIT call with PSB in program
- EZEDLPCB special function word reference
- GSAM file association

Run-time JCL templates for IMS BMP: Table 21 on page 77 shows the JCL templates and procedures used for running IMS BMP programs. The templates and procedures are selected based on the following information:

- The environment
- Whether the program is a main or called program
- Use of SQL in the program

Note: DL/I is always present in the IMS BMP environment.

Table 21. Templates for running IMS BMP

Environment	Program	SQL	Template	Procedure called by template
IMS BMP	Main	Yes	EFK2MEIA	IMSBATCH
IMS BMP	Main	No	EFK2MEIB	IMSBATCH
IMS BMP	Called	*	EFK2MEBA	

* The use of the template is not affected by SQL usage.

Each of the JCL templates contains a ?DD? line that starts in column one. This line indicates where the generation function is to insert the DD statements for files and databases used by the program. Move this line within the JCL template to control where the DD statements are placed. If you remove the line that contains ?DD?, the DD statements are not included in the generated JCL.

Run-time JCL templates for VSE batch: Table 22 shows the JCL templates used for running VSE programs. The templates are selected based on the following information:

- The environment
- Whether the program is a main or called program
- Use of SQL in the program
- Use of DL/I in the program

Note: Procedures are not called by JCL templates for running VSE programs.

Table 22. JCL templates used for running VSE programs.

Environment	Program	SQL	DL/I	Template
VSE batch	Main	Yes	Yes	EFK2VEBC
VSE batch	Main	Yes	No	EFK2VEBD
VSE batch	Main	No	Yes	EFK2VEBB
VSE batch	Main	No	No	EFK2VEBE
VSE batch	Called	*	*	EFK2VEBA

*The use of the template is not affected by SQL or DL/I usage.

Each of the JCL templates contains a ?DLBL? line that starts in column one. This line indicates where the generation function is to insert the DLBL statements for files and databases used by the program. Move this line within

the JCL template to control where the DLBL statements are placed. If you remove the line that contains ?DLBL?, the DLBL statements are not included in the generated JCL.

EFK2VEBA

For called programs, only DLBL statements are generated. These statements can be incorporated into the sample run-time JCL of any main program that calls the program.

More about DL/I usage for VSE batch: In the VSE batch environment, DL/I usage is indicated if the program includes at least one of the following attributes or functions:

- Database PCB, other than for the ELAWORK or ELAMSG synonyms, included in the PSB
- EZEDLPCB special function word passed on a CALL statement
- EZEDLPCB or EZEDLPSB special function word received as a parameter for a called program
- CSPTDLI call
- EZEDLPCB special function word reference

REXX templates for VM

Table 23 shows the REXX templates used for running VM programs. The templates are selected based on the following information:

- The environment
- Whether the program is a main or called program
- Whether SQL is used by the program

Table 23. REXX templates used for running VM programs.

Environment	Program	Template	EXECs called by template
VM CMS	Main	EFK2CERB	ELASETUP
VM batch	Called	EFK2CERA	ELARUN ELACLEAN

Templates EFK2CERA and EFK2CERB contain a ?ALLOC? line that starts in column one. This line indicates where the generation function is to insert calls to ELACALLC EXEC to do the FILEDEF and DLBL commands for files used by the programs. Move this line within the templates to control where ELACALLC EXEC statements are placed. If you remove the line that contains ?ALLOC?, the calls to ELACALLC EXEC are not included in the generated run-time exec.

CLIST templates for MVS/TSO

Table 24 shows the CLIST templates and procedures used for MVS/TSO run time. The templates and procedures are selected based on the following information:

- The environment
- Whether the program is a main or called program
- Use of SQL in the program
- Use of DL/I in the program

Table 24. CLIST templates and procedures used for MVS/TSO run time

Environment	Program	SQL	DL/I	Template	CLIST called by template
MVS/TSO	Called	Yes	No	EFK2META	
MVS/TSO	Called	No	*	EFK2META	
MVS/TSO	Main	Yes	No	EFK2METC	ELATALC
MVS/TSO	Main	No	Yes	EFK2METB	ELATALC
MVS/TSO	Main	No	*	EFK2METD	ELATALC

*The use of the template is not affected by DL/I usage.

Each of the CLIST templates contains an ?ALLOC? line that starts in column one. This line indicates where the generation function is to add the allocate (ALLOC) commands for files and databases used by the program. Move this line within the CLIST template to control where the commands are placed. If you remove the line that contains ?ALLOC?, the ALLOC commands are not included in the generated CLIST procedures.

For a called program, only ALLOC commands are generated. The CLIST for the called program can be run from the CLIST for the calling program.

More about SQL and DL/I use: In the MVS/TSO environment, DL/I usage is indicated if the program includes at least one of the following attributes or functions:

- Database PCB, other than for ELAWORK or ELAMSG, is included in the PSB
- EZEDLPCB passed on a CALL statement
- EZEDLPCB or EZEDLPSB received as a parameter for a called program
- CSPTDLI call
- EZEDLPCB reference

EFK2META

For called programs, only ALLOC commands are generated. The CLIST for the called program can be run by the CLIST for the calling program.

File and database allocation templates

Some of the files or databases used by a program require file or database templates for generating file allocation statements. Table 25 shows the template selected based on the following information:

- The type of record organization
- Whether the record is input or output
- The associated file types

Table 25. Templates for generating file allocation statements

Record organization	Input or output	Associated file type	CLIST template for MVS/TSO	JCL template for MVS batch or IMS BMP	JCL template for VSE batch	REXX EXEC template for VM
DL/I segment	Input or output		EFK2MTDL	EFK2MDLI	EFK2VDLI	
Indexed Relative Serial	Input	VSAM, VSAMRS	EFK2MTVI	EFK2MVISI	EFK2VVSI	EFK2CVVI
Indexed Relative Serial	Output	VSAM, VSAMRS	EFK2MTVO	EFK2MVSO	EFK2VVSO	EFK2CVVO
Serial	Input	SEQ, SEQRS	EFK2MTSI	EFK2MSDI	EFK2VSEI	EFK2CVSI
Serial	Output	SEQ, SEQRS	EFK2MTSO	EFK2MSDO	EFK2VSEO	EFK2CVSO
Serial	Input	GSAM		EFK2MGSI, EFK2MIMS		
Serial	Output	GSAM		EFK2MGSO, EFK2MIMS		

Note: The statements generated from file and database templates are included at points marked by the ?DD?, ?ALLOC?, or ?DLBL? lines in the program template.

EFK2VDLI

The EFK2VDLI template has the DLBL statement commented out. The VisualAge Generator Developer does not collect the resource association information needed to build this statement. You must provide the final tailoring of these DLBL statements in the sample run-time JCL after the program has been generated.

EFK2MTDL

The EFK2MTDL template has the ALLOC command commented out. The VisualAge Generator Developer does not collect the resource association information needed to build this statement. You must provide the final tailoring of these ALLOC commands in the sample CLIST after the program has been generated.

EFK2MDLI

The EFK2MDLI template is only included if the target environment is MVS batch. For the IMS BMP target environment, the database should already be allocated to the IMS control region.

This template has the DD statement commented out. The VisualAge Generator Developer does not collect the resource association information needed to build this statement. You must provide the final tailoring of these DD statements in the sample run-time JCL after the program has been generated.

EFK2MIMS

The EFK2MIMS template is an extra template for GSAM. For IMS BMP jobs, if any of the serial files is associated with a GSAM file, the template EFK2MIMS is also included.

File and database allocation placeholder templates

The generated sample run-time files for programs must be modified to include DD statements, DLBL statements, ALLOC, or FILEDEF commands for files referenced using the EZEDEST or EZEDESTP special function words, or for files and databases used by programs that are called or transferred-to from the base program.

Table 26 shows the templates that generate comments in the sample run-time files to indicate the place where DD statements, DLBL statements, ALLOC, or FILEDEF commands can be included.

Table 26. Templates that generate comments in sample run-time files

Program function	CLIST template for MVS/TSO	JCL template		
		for MVS batch or IMS BMP	JCL template for VSE batch	REXX EXEC template for VM
Program uses an XFER or DXFR statement with EZEAPP	EFK2MTEA	EFK2MEZA	EFK2VEZA	EFK2CVEA

Table 26. Templates that generate comments in sample run-time files (continued)

Program function	CLIST template for MVS/TSO	JCL template for MVS batch or IMS BMP	JCL template for VSE batch	REXX EXEC template for VM
Program uses a CALL, XFER, or DXFR statement to a specific program, or the /RT generation option indicates a transfer to a specific program	EFK2MTCL, page 82	EFK2MCAL	EFK2VCAL	EFK2CVCL, page 82
Program sets the EZEDEST or EZEDESTP special function word	EFK2MTED	EFK2MEZD	EFK2VEZD	EFK2CVED

EFK2MTCL

The EFK2MTCL template generates an EXEC statement to run the CLIST generated for the called or transferred-to program. The CLIST for the base program does not have to be modified to include the ALLOC commands for called or transferred-to programs.

EFK2CVCL

The EFK2CVCL template generates an EXEC statement to run the run-time REXX exec generated for the called or transferred-to program. The run-time REXX exec for the base program does not have to be modified to include the FILEDEF or DLBL commands for called or transferred-to programs.

Modifying templates

During the installation process, the default templates are moved to C:\Program Files\Vast\bin for the Smalltalk version of VisualAge Generator or C:\IBMVJava\IDE\program for the Java version.

Templates can be modified to meet installation requirements for preparing and running programs. These modifications are usually done by a system administrator or project leader. See “Reasons to modify templates” on page 83 for detailed reasons to modify templates.

To modify the templates, create a new directory and copy all the templates from the default templates directory (C:\Program Files\Vast\bin for the Smalltalk version of VisualAge Generator or C:\IBMVJava\IDE\program for the Java version). Because different projects can require the use of different templates, set up separate template directories for each project. Each directory must contain copies of all the templates. Make your changes to these copies of the templates. To use templates that you have placed in directories other than

the default templates directory, specify the directory name using the /TEMPLATES generation option. Use a standard editor to modify the templates.

User-defined symbolic parameters can also be inserted in templates. The values for user-defined symbols are specified at generation time using the /SYMPARM generation option. When you use symbolic parameters in a template for MVS, VSE, or VM programs, ensure that the generated statement does not go past column 71 when the largest possible values are substituted for the symbol. Statements longer than 71 columns in host templates are truncated. The length restriction for templates does not apply for OS/2 programs. Also, be sure your changes result in the generation of valid syntax. The generation facility does not check the syntax.

If you modify the templates, you might need to modify the corresponding cataloged procedures. Conversely, if you modify the cataloged procedures, you might need to modify the corresponding templates. Cataloged procedures are shipped with VisualAge Generator Server for MVS, VSE, and VM and installed on the target system where preparation is to take place. Cataloged procedures can also be tailored. The system administrator can change the name of the cataloged procedure being started, assign parameter values, and override DD statements or ALLOC commands.

If you modify the templates for the VM environment, you might need to modify the corresponding REXX execs provided with VisualAge Generator Server for MVS, VSE, and VM. Conversely, if you modify the REXX execs provided with the VisualAge Generator Server for MVS, VSE, and VM, you might need to modify the corresponding templates for the VM environment. The REXX execs are shipped with VisualAge Generator Server for MVS, VSE, and VM and installed on the target system where preparation is to take place. The REXX execs provided with VisualAge Generator Server for MVS, VSE, and VM can also be tailored. The system administrator can change the name of the REXX execs being started, assign parameter values, and override FILEDEF and DLBL commands.

See “Examples of modifying templates” on page 106 for examples showing how to modify templates.

Reasons to modify templates

You do not need to modify templates often. If changes need to be made to a template, your system administrator or project leader usually makes them. Some reasons why templates might need to be modified include these:

- Implementing installation data set naming conventions
- Changing the name of the cataloged procedure being called
- Changing the preparation process

- Assigning parameter values
- Adding DD statements to the STEPLIB concatenation
- Overriding DD statements
- Overriding ALLOC statements
- Overriding DLBL statements
- Overriding FILEDEF statements

Additional reasons for modifying templates include the following:

- Adding override DD statements for S1.SYSPRINT to the EFK2MMSU and EFK2MMST JCL templates to prevent printing the MFS listings
- Adding the OWNER or EXPLAIN option to templates EFK2MBDA, EFK2MBDB, EFK2MBDC, or EFK2MBDD
- Adding or changing the REGION parameter for a step
- Including a JES2 JOBPARM or JES3 //*MAIN statement after the JOB statement in the JOB statement file to change the PROCLIB referenced when the preparation job runs
- Adding a JES3 //*MAIN SYSTEM=xx statement, where xx is the name of the target system, after the JOB statement in the JOB statement file to direct a preparation job to a specific system
- Adding the XDEST parameter on the VSE/POWER JOB statement to direct a preparation job to a specific system
- Adding symbols for AMODE and RMODE in the link-edit steps of the templates. Specify AMODE(31) and RMODE(ANY) for most generated programs; use AMODE(24) and RMODE(24) only if the generated program must be statically linked with a non-VisualAge Generator program that requires AMODE(24) and RMODE(24).
- Adding additional parameters on the call to ELAPREP in the generated preparation REXX execs for VM

Modifying templates and procedures for MVS environments

Preparation templates for MVS systems run cataloged procedures to perform program preparation. You can tailor the procedures to meet your installation's requirements.

The default procedures are shipped with VisualAge Generator Server for MVS, VSE, and VM. If you change these procedures, you might need to make changes in the corresponding templates.

Table 27 on page 85 shows the naming convention used for the preparation procedures, with the exception of ELARLINK.

Table 27. Procedure naming convention

Text	Description
ELA	The prefix for all procedures
P	DB2 precompile
T	Translate
C	COBOL compile
L	Link
B	DB2 bind

Preparing programs on a single system

Table 28 shows the procedures run by the generated preparation JCL. If you are preparing the program on an MVS environment, you can use these procedures. If you change these procedures, you might need to make changes in the corresponding JCL templates.

If you use ANSI SQL instead of DB2, modify the procedures to use your ANSI SQL processor, if any.

Table 28. Procedures for preparation JCL for an MVS environment

Procedure	Purpose
ELAB	DB2 bind. This procedure is used for moving a program to a second system.
ELACL	COBOL compile and link
ELACLB	COBOL compile, link, and DB2 bind. This procedure is used for programs that do not have any SQL statements, but have specified a DB2 work database for the IMS/VS environment.
ELAL	Link-edit the map group format module
ELAPCLB	DB2 precompile, COBOL compile, link, and DB2 bind
ELAPTCLB	DB2 precompile, CICS translate, COBOL compile, link, and DB2 bind
ELARLINK	Relink modules that require static linkage
ELATCL	CICS translate, COBOL compile, and link

Preparing programs on different systems

Table 29 on page 86 shows the preparation procedures you can use when you compile on a development system and link on a different production system, or generate on one system and compile and link on another.

Table 29. Preparation procedures for preparing on different systems

Procedure	Purpose
ELAC	COBOL compile
ELAL	Link only
ELALB	Link and DB2 bind
ELAPC	DB2 precompile and COBOL compile
ELAPTC	DB2 precompile, CICS translate, and COBOL compile
ELATC	CICS translate and COBOL compile

Procedure name and symbolic parameters

If you modify the procedure name, be sure to change the JCL templates used to run the procedure to use the new name.

If you modify the symbolic parameters defined in the procedure, be sure to change the symbolic parameters in the JCL templates that use the procedure. If you add or delete symbolic parameters for a procedure, review the JCL templates that run the procedure to determine if you need to change the templates.

See “Chapter 12. Symbolic parameters” on page 123 for more information on symbolic parameters.

Required DB2 options

The following options are required for DB2 usage. These options are included in the cataloged procedures. Do not remove these options:

- HOST(COB2)
- APOSTSQL
- QUOTE

CICS-required conversion options

The following options are required for CICS conversion. These options are included in the cataloged procedures. Do not remove these options:

- COBOL2
- NOSEQ
- QUOTE

CICS/ESA Version 3 Release 2 or higher requires the SP translator option.

DBCS translator option for CICS/ESA

Double-byte character set (DBCS) or mixed data requires the DBCS translator option and CICS/ESA Version 3 Release 1 Modification 1 or higher. The DBCS option is included in the preparation procedures based on the value of the EZEDBCS symbolic parameter.

Setting COBOL run-time options for MVS

You might need to set additional COBOL run-time options, depending on the run-time library in use and the options your program requires.

LE run-time options: The following run-time options are required when you use the LE run-time library:

- STORAGE(00,...)
- CBLPSHPOP(ON).

For the DATA(31) COBOL option, specify the ANYWHERE suboption for the HEAP run-time option. Otherwise, all COBOL and VisualAge Generator Server for MVS, VSE, and VM storage is allocated with a 24-bit address.

The TRAP and TERMTHDACT run-time options control error condition handling. The LE documentation indicates TRAP(OFF) must be specified if programs use CBLTDLI or ASMTDLI interfaces. CBLTDLI and ASMTDLI are used by generated programs and host services.

TERMTHDACT controls the amount of information reported on a severe error detected by TERMTHDACT. TERMTHDACT(DUMP) produces an LE dump in addition to an LE message.

You can either set the LE installation defaults to these values or modify the link-edit templates to link in program-specific options in CSECT CEEUOPT.

Refer to the *COBOL Programming Guide* (SC26-4767) for more information on specifying LE run-time options.

Modifying LE user exits: In addition to setting appropriate LE run-time options, you can modify the LE assembler user exit (CEEBXITA) to request an ABEND instead of an error return for all abend codes for programs that can have updates to databases or for recoverable files that have not been committed. An ABEND forces a rollback; a return with an error code commits updates to the database.

To use a different copy of the user exit for different programs, you can modify the link-edit templates to link the user exit with the program load module.

Refer to the *LE Programming Guide* (SC26-4818) for more information on condition handling and the assembler user exit.

COBOL compiler options for MVS

You might need to set additional COBOL run-time options, depending on the run-time library in use and the options your program requires.

Required options: The preparation procedures include the required compiler options. These options do not have to match the default options at your installation. Do not remove these options from the preparation procedures.

The following are the required COBOL compiler options:

- NODYNAM
- RENT
- RES
- TRUNC(BIN)
- NUMPROC(NOPFD)
- NOSEQ
- LIB
- QUOTE
- NOCMR2

NODYNAM compiler option: The NODYNAM compiler option is required to enable static linking of the VisualAge Generator Server for MVS, VSE, and VM interface stub program with the generated program. Calls to other programs can be dynamic depending on whether the COBOL call is generated as a CALL IDENTIFIER or CALL LITERAL. CALL IDENTIFIER and CALL LITERAL calls are used as follows:

CALL IDENTIFIER

This dynamic call is generated by VisualAge Generator Developer for a CALL or DXFR statement when you use the default linkage in non-CICS environments. CALL IDENTIFIER is also used for calls to mapping services programs and table programs. This enables printer map groups and tables to be generated independent of the programs that use them.

CALL LITERAL

This is either a dynamic or static call based on the compiler option used. VisualAge Generator Developer generates this type of call for a CALL or DXFR statement when you request static linkage. The VisualAge Generator Developer also generates this type of call for calls to VisualAge Generator Server for MVS, VSE, and VM modules. The statically linked VisualAge Generator Server for MVS, VSE, and VM modules contain the following:

- A few modules that are used frequently
- A stub module that dynamically links to the major run-time functions

VisualAge Generator requires that the NODYNAM compiler option be specified so that all CALL LITERAL statements are treated as static calls.

DBCS compiler option: If you use double-byte character set (DBCS) or mixed data, the DBCS compiler option is required. The DBCS option is included in the preparation procedures based on the value of the EZEDBCS symbolic parameter.

RES compiler option: The RES option can be omitted for COBOL because COBOL supports only the RES option function.

Included compiler options: Several COBOL compiler options might be useful to you. The following options are included in the preparation procedures, but are not required. You can remove these options if you do not need them.

OFFSET

Produces a cross-reference listing of the hexadecimal offsets in the module and the COBOL statement number. This is recommended to help you while you are debugging your program.

OPTIMIZE

Can significantly increase the compile time performance, but might also give faster run-time performance. Therefore, it might be beneficial to use the NOOPTIMIZE option during testing and the OPTIMIZE option when moving the program to production.

Additional compiler options: Other compiler options that are not included in the cataloged procedures that you might find useful follow:

TEST The use of the TEST compiler option varies depending on the COBOL debug facility you use.

- **COBOL Debug Facility**

If you plan to test your program using the COBOL debug facility before putting the program into production, you can specify the TEST compiler option. TEST forces the NOFDUMP option. Before putting the program into production, compile again with the NOTEST option. Using the NOTEST option reduces the load module size.

- **CODE/370; Debugger**

If you plan to test your COBOL program using the CODE/370; Debugger before putting the program into production, you can specify the TEST compiler option. Before putting a program into production, compile again with the TEST(NONE,SYM) or NOTEST option. LE provides a formatted dump with either option. Specify the SYM suboption of the TEST compiler option to have symbolic variables included in the formatted dump.

NOFDUMP

Using NOFDUMP prevents COBOL from giving a formatted dump on an abend. However, specifying FDUMP causes significantly larger load modules. You must determine what is important to you—better performance due to smaller load modules, or the additional information provided if the program is not successful.

SSRANGE

Specify SSRANGE if you want run-time subscript checking. NOSSRANGE eliminates run-time subscript checking. Because using NOSSRANGE can result in better performance, specify NOSSRANGE when you move the program to production.

DATA(xx)

The VisualAge Generator Developer /DATA generation option enables you to specify whether the preparation JCL uses the DATA(31) or the DATA(24) compiler option. This option controls whether data areas are acquired above or below the 16MB (MB equals 1048576 bytes) boundary.

DATA(31) is the default for CICS programs. DATA(24) is the default for non-CICS programs. The preparation procedures cause the generated programs to be linked with AMODE(31), RMODE(ANY) for all environments.

For CICS for MVS/ESA and CICS for VSE/ESA, you can only specify 24-bit addressing if the dynamic storage required by the generated COBOL program, mapping services program, or table is less than 64KB.

Use the following rules when generating or linking programs, tables, and map groups:

- Specify 24-bit addressing for the following:
 - Any program that calls a program or program linked AMODE(24)
 - The first program in the run unit if any generated program in the run unit is linked AMODE(24) or if any program that uses DL/I is generated with /DATA=24
 - Tables and map groups if any program that uses the table is linked AMODE(24)
 - Any DL/I program in VSE batch or in non-CICS environments on MVS if IMS/ESA is not installed
- If you specify 31-bit addressing for DL/I programs in the MVS batch or MVS/TSO environments, you must install IMS/ESA and you must specify the VisualAge Generator Server for MVS, VSE, and VM IMSESA installation option as IMSESA='Y'. Refer to the

VisualAge Generator Server Guide for MVS, VSE, and VM for information about the IMSESA installation option.

Refer to your COBOL documentation for more information about the compiler options.

Compiler options that are not supported: The compiler option NAME is not supported.

Modifying the MVS JOB card

Figure 15 shows the default JOB statement for MVS.

```
//%EZEJOB% JOB (MYACCT),CLASS=A,MSGCLASS=A,NOTIFY=%EZEUSRID%
```

Figure 15. Default JOB statement for MVS

You can change the JOB statement file to use your own symbolic parameters for the following:

- JOB name
- Accounting information
- Class
- Notify ID

Refer to the *VisualAge Generator Installation Guide* for information about changing the notify value using the EZEUSRID environment variable and for information about modifying values in the hpt.ini file.

To minimize the amount of overriding you must do, the system administrator or project leader can create different JOB statements or modify the default JOB statement to meet the standards or requirements of your installation.

Other modifications that you can make to the JOB statement in the preparation JCL include changing the PROCLIB and directing the job to run on a specific system.

Changing the PROCLIB: Figure 16 on page 92 shows an example of what you can add to the statement following the JOB statement to use a different library when the job runs.

```
JES2 JOBPARM
```

or

```
JES3 //*MAIN
```

Figure 16. Changing the PROCLIB

Directing the job to run on a specific system: You might want the preparation job to run on a specific system. For example, you might have the COBOL compiler and DB2 installed on only one system. Figure 17 shows the statement you can add after the JOB statement to direct the preparation job to the particular system.

```
//*MAIN SYSTEM=xx
```

Figure 17. Directing to a single system

Modifying templates for the OS/400 environment

You can modify the following templates:

- EFK24PCL
- EFK24PMN
- EFK24PSC
- EFK24PSM
- EFK24TCM
- EFK24TAM
- EFK24PEJ
- EFK24EBC

Do not modify the following templates:

- EFK24PBC
- EFK24PEC
- EFK24TEM
- EFK24TMJ
- EFK24EEC

You can modify the OS/400 preparation command file template to do the following:

- Delete source and intermediate files
- Change the default drive for the shared folder
- Suppress the transfer messages

Refer to the prologues and comments in the templates for more information about changing templates in the OS/400 environment.

Modifying templates for VSE environments

Preparation templates for VSE run the following procedure to perform program preparation:

ELACUSTP

This procedure sets up the program user libraries, product libraries, and SQL preprocessor options.

If you change this procedure, you might need to make changes in the corresponding templates.

An additional procedure is run if you specify single program user mode for preparing DB2/VSE programs. Programs that set the value of the SQLSTMDE symbolic parameter to SINGLEUSER must also customize the procedure ELASQLDB, which starts the DB2/VSE database.

SQL preprocessing for VSE

When a generated SQL program is prepared for running on VSE, an SQL preprocessor step is included in the preparation JCL. The preprocessor step creates a package (access module) that is stored in the DB2/VSE database.

The templates are used to generate the preprocessor step. See any of the following for more information on which templates are used, based on environment and database usage:

- “Program templates for CICS for VSE/ESA and VSE batch preparation” on page 64
- “Map group templates for CICS for VSE/ESA and VSE batch preparation” on page 66
- “Run-time JCL templates for VSE batch” on page 77

The project leader determines a set of standard preprocessor options that are needed for a project. Changes to options can be made in the default processor options file for VSE preparation, ELASQLPR, which is stored in the host services library. The preparation process can also be directed to use a different options file by setting the value of the symbolic parameter SQLPROPT to the new options file name.

Starting SQL preprocessing mode: The default file containing DB2/VSE startup options is named ELASQLST. ELASQLST is stored in the host services sublibrary, PRD2.EZELIB. If you want to specify a different DB2/VSE startup options file, use the SQLSTOPT symbolic parameter. If the new DB2/VSE startup options file is stored in a sublibrary other than the host services sublibrary, use the VUSERLIB symbolic parameter to specify the sublibrary.

Figure 18 shows how to change the DB2/VSE startup options file to MYSOQLST. The file MYSOQLST is stored in sublibrary PRD5.OBJLIB.

```
/SYMPARM=SQLSTOPT,'MYSOQLST'  
/SYMPARM=VUSERLIB,'PRD5.OBJLIB'
```

Figure 18. Changing the DB2/VSE startup options file

The SQL preprocessor step can be started in DB2/VSE multiple program user mode or single program user mode. The default template for preparation runs in DB2/VSE multiple program user mode. Multiple program user mode and single program user mode are used as follows:

- Multiple program user mode is used if the relational database is shared among CICS users or batch jobs, or both. The system operator must have started DB2/VSE before a job can run in multiple program user mode.
- Single program user mode is used if no other CICS transactions or batch jobs need to access the relational database while the job runs. In single program user mode, the job control starts DB2/VSE and starts the DB2/VSE preprocessor.

If you plan to use single program user mode, you must modify the procedure ELASOQLDB. The preparation process uses this procedure to start the DB2/VSE database.

To change from multiple program user mode to single program user mode, use the SQLSTMDE symbolic parameter to change the value.

Figure 19 shows how to change the DB2/VSE startup mode to single program user mode.

```
/SYMPARM=SQLSTMDE,'SINGLEUSER'
```

Figure 19. Changing the DB2/VSE startup mode

Required DB2/VSE options: The following options are required for DB2/VSE usage. These options are included in ELQSOQLPR.A, which is shipped with VisualAge Generator Server for MVS, VSE, and VM and stored in the host services sublibrary, PRD2.EZELIB. Do not remove these options:

- COB2
- QUOTE

Setting additional SQL preprocessor options: Review the SQL preprocessor options in the DB2/VSE program programming document for your version of DB2/VSE. You can add the following two parameters to the default preprocessor parameters. These parameters are specified in the file

ELASQLPR, which is shipped with VisualAge Generator Server for MVS, VSE, and VM and stored in the host services library.

ISQL(CS)

Cursor stability isolation level

EXPLAIN(YES)

Places information about the structure and run-time performance for a DELETE, INSERT, or SELECT statement into one or more user-defined explain tables. The EXPLAIN preprocessor option is available for DB2/VSE version 3.4 or higher.

Naming DB2/VSE program packages: The default package name for VSE DB2/VSE program packages is the program name. If you want to specify a different package name, use the SQLPKGNM symbolic parameter to change the package name.

Figure 20 shows an entry made to a generation options file to set the value of the SQLPKGNM symbolic parameter to NEWPGM. The resulting DB2/VSE package name is NEWPGM.

```
/SYMPARM=SQLPKGNM, 'NEWPGM'
```

Figure 20. Changing the SQLPKGNM symbolic parameter

Setting COBOL run-time options for VSE

You might need to set additional COBOL run-time options, depending on the run-time library in use and the options your program requires.

See any of the following for information about compiler options:

- “Required options” on page 88
- “Included compiler options” on page 89
- “Additional compiler options” on page 89
- “Compiler options that are not supported” on page 91

Modifying the VSE JOB card

Figure 21 shows the default JOB statement for VSE.

```
* $$ JOB JNM=%EZEGMBR%,CLASS=%PWRCLAS,0%,DISP=D,NTFY=(%NODUSRID,YES%)
```

Figure 21. The default JOB statement for VSE

You can change the JOB statement file to use your own symbolic parameters for the following:

- JOB name
- Class
- Notify ID

To minimize the amount of overriding you must do, the system administrator or project leader can create different JOB statements or modify the default JOB statement to meet the standards or requirements of your installation.

You might want to make some of the following modifications to the JOB statement in the preparation JCL.

Changing the PROC library name: Figure 22 shows the statement that you might change.

```
// LIBDEF PROC,SEARCH=
```

Figure 22. Library name statement

Figure 23 shows a symbolic parameter definition you can add to your generation options that changes the library name.

```
/SYMPARM=PROCLIB,'library name'
```

Figure 23. Changing the library name using a symbolic parameter

Directing the job to run on a specific system: You might need to run the preparation job on a specific VSE system. For example, if you have the COBOL compiler and DB2/VSE installed on only one system, you can specify the following statements, depending on whether the destination system changes:

- If the destination system is not likely to change, you can add the XDEST=xx parm to your VSE/POWER JOB statement to direct the job to a specific system. Figure 24 shows an example of this statement, where xx is the destination system node name.

```
* $$ JOB JNM=TEST,XDEST=xx
```

Figure 24. Setting the destination system

- If the destination system might change, you can use a symbolic parameter to specify the destination. This enables you to override the specified value during generation. Figure 25 shows an example of using a symbolic parameter in the JOB card to set the destination system.

```
* $$ JOB JNM=TEST,XDEST=%NODED%
```

Figure 25. Using a symbolic parameter to set the destination system in the JOB card

Figure 26 shows how you then set the symbolic parameter in the generations options defaults file.

```
/SYMPARM=NODED, 'VSESYS1'
```

Figure 26. Setting symbolic parameter

This parameter directs the job to run on system VSESYS1. If you want to direct the job to a different system, you can define a new value for the symbolic parameter NODED. You do not need to change the JOB card.

Modifying preparation templates and EXECs for VM environments

Overview of program preparation for VM

The VisualAge Generator generates a preparation REXX file for each program. It contains a preparation step for each object that needs to be prepared for execution in the VM environment (see “Chapter 13. Outputs of COBOL generation” on page 137 for information about the types of objects that are generated). The preparation REXX file, which is created from the templates that are described in “Preparation REXX for VM” on page 70, is transferred to VM along with other files that it needs to prepare the object.

The preparation REXX calls the following execs that are shipped with VisualAge Generator Server for MVS, VSE, and VM.

- The ELASETUP EXEC is called to link and access the minidisk or SFS directory that contains the VisualAge Generator Server for MVS, VSE, and VM code.
- The ELACLEAN EXEC is called to release and detach the minidisk or SFS directory that contains the VisualAge Generator Server for MVS, VSE, and VM code.
- The ELAPREP EXEC is called for each object that needs to be processed.

ELAPREP processes each object differently based on the parameters that are passed to it. For programs and tables that are COBOL source code, the COBOL compiler and the CMS LKED command are used to process the object. If the program contains SQL statements, ELAPREP calls ELASQLAA EXEC to perform SQL/DS VM preprocessing. For map group format modules, only the CMS LKED command is called to link-edit the object.

Because the preparation REXX exec is generated from templates, you can modify the templates in order to customize the preparation REXX exec.

ELAPREP parameters

The ELAPREP EXEC supports many keyword parameters that control which part it processes and how that part is processed. The preparation REXX exec that is generated sets some of the ELAPREP parameters specifically for the

part that is being generated. Do not change the value of these parameters: MBR=, ENV=, COBOLTYPE=, DBCS=, PARTTYPE=, and SQL=.

ELAPREP uses the installation variable information that was set during VisualAge Generator Server for MVS, VSE, and VM installation. ELAPREP must know the location of the COBOL compiler and names of maclibs and txtlibs. Some ELAPREP parameters allow you to override installation variables. Do this carefully. The following ELAPREP parameters can be used to override installation variables: COBLIB=, COBLOC=, COBLKED=, COBLKEDLOC=, SQLLOC=, ELAMACLIB=, ELAMACLOC=, ELALOADLIB=, and ELALOADLOC=.

Other ELAPREP parameters can be changed or added to the generated preparation REXX exec. These ELAPREP parameters can be added to the templates so that they are included whenever a program is generated. The following sections discuss ELAPREP parameters that can be used to control the preparation process.

COBOL compiler options for VM

ELAPREP calls the COBOL compiler with options that satisfy most objects and environments. ELAPREP uses the installation variable information that was set during VisualAge Generator Server for MVS, VSE, and VM installation to determine which COBOL compiler to use and where it is located. This section discusses the compiler options that ELAPREP uses and options that you might want to change.

Required options: ELAPREP uses some compiler options that are required by VisualAge Generator. These options do not have to match the default options at your installation. Do not remove these options:

- NODYNAM
- RENT
- RES
- TRUNC(BIN)
- NUMPROC(NOPFD)
- NOSEQ
- LIB
- QUOTE
- NOCMR2

NODYNAM

The NODYNAM compiler option is required to enable static linking of the VisualAge Generator Server for MVS, VSE, and VM interface stub program with the generated program. Calls to other programs can be dynamic depending on whether the COBOL call is generated as a CALL IDENTIFIER or CALL LITERAL. CALL IDENTIFIER and CALL LITERAL calls are used as follows:

CALL IDENTIFIER

This dynamic call is generated by VisualAge Generator for a CALL or DXFR statement when you use the default linkage. CALL IDENTIFIER is also used for calls to mapping services programs and table programs. This enables printer map groups and tables to be generated independent of the programs that use them.

CALL LITERAL

This is either a dynamic or static call based on the compiler option used. VisualAge Generator generates this type of call for a CALL or DXFR statement when you request static linkage. It also generates this type of call for calls to VisualAge Generator Server for MVS, VSE, and VM modules.

The statically linked VisualAge Generator Server for MVS, VSE, and VM modules contain the following:

- A few modules that are used frequently
- A stub module that dynamically links to the major run-time functions

VisualAge Generator requires that the NODYNAM compiler option be specified so that all CALL LITERAL statements are treated as static calls.

RES The RES option is required if COBOL is being used. ELAPREP determines if you are using COBOL and includes the RES option if necessary by using the installation variable information that was set during VisualAge Generator Server for MVS, VSE, and VM installation.

Optional compiler options: ELAPREP uses several COBOL compiler options that can be changed based on your requirements. ELAPREP can be passed additional parameters to control which compiler options are used. Unless otherwise noted, you can modify the VisualAge Generator templates to automatically include these parameters or edit the generated preparation REXX exec.

DISK/PRINT/NOPRINT

The ELAPREP COBLIST= parameter can be used to control the generation of a program listing from the COBOL compiler. COBLIST=NOPRINT is the default and suppresses the generation of any listing. COBLIST=DISK generates a listing to the same file mode that contains the COBOL source. COBLIST=PRINT generates a listing to your virtual printer. The user-defined symbolic parameter COBLIST can be used to specify the value of the ELAPREP COBLIST= parameter.

LIST/OFFSET

The COBOL compiler option OFFSET includes a condensed procedure division listing, global tables, literal pools, and information about working-storage that is used by your program. The COBOL compiler option LIST includes an assembler-language expansion of your source code in addition to the items produced by the OFFSET option. Specify the ELAPREP COBASM=Y parameter if you want the COBOL compiler option LIST to be used. By default, ELAPREP uses COBASM=N, which causes the OFFSET compiler option to be used.

DBCS If you use double-byte character set (DBCS) or mixed data, the DBCS compiler option is required. Parameter DBCS=Y is automatically included on the call to ELAPREP if the generator determined your program used DBCS or mixed data.

DATA(XX)

The VisualAge Generator /DATA generation option enables you to specify whether the preparation exec uses the DATA(31) or the DATA(24) compiler option. This option controls whether data areas are acquired above or below the 16MB (a megabyte (MB) equals 1,048,576 bytes) boundary. DATA(31) is the default.

OPTIMIZE/NOOPTIMIZE NOOPTIMIZE can significantly increase the compile time performance, but OPTIMIZE might give faster run-time performance. Therefore, it might be beneficial to use the NOOPTIMIZE option during testing, and use the OPTIMIZE option when moving the program to production. Specify the ELAPREP COBOPT=Y parameter if you want the COBOL compiler option OPTIMIZE to be used. By default, ELAPREP uses COBOPT=N, which causes the NOOPTIMIZE compiler option to be used.

TEST The use of the TEST compiler option varies depending on the COBOL debug facility you use.

- **COBOL DEBUG FACILITY**

If you plan to test your program using the COBOL debug facility before putting the program into production, you can specify the TEST compiler option. TEST forces the NOFDUMP option. Before putting the program into production, compile again with the NOTEST option. Using the NOTEST option reduces the load module size.

Specify the ELAPREP COBTEST=Y parameter if you want the COBOL compiler option TEST to be used. By default, ELAPREP uses COBTEST=N, which causes the NOTEST compiler option to be used. You can modify the template to specify the value of this ELAPREP parameter.

- **CODE/370; Debugger**

If you plan to test your COBOL program using the CODE/370; Debugger before putting the program into production, you can specify the TEST compiler option. Before putting a program into production, compile again with the TEST(NONE,SYM) or NOTEST option. LE provides a formatted dump with either option. Specify the SYM suboption of the TEST compiler option to have symbolic variables included in the formatted dump.

Specify the ELAPREP COBTEST=Y parameter if you want the COBOL compiler option TEST(ALL,SYM) to be used. By default, ELAPREP uses COBTEST=N, which causes the TEST(NONE,SYM) compiler option to be used. You can modify the template to specify the value of this ELAPREP parameter.

Additional compiler options: Other compiler options that are not included in ELAPREP that you might find useful are:

NOFDUMP/FDUMP

Using NOFDUMP prevents COBOL from giving a formatted dump on an abend. However, specifying FDUMP causes significantly larger load modules. You must determine what is important to you—better performance due to smaller load modules, or the additional information provided if the program is not successful.

SSRANGE/NOSSRANGE

Specify SSRANGE if you want run-time subscript checking. NOSSRANGE eliminates run-time subscript checking. Because using NOSSRANGE can result in better performance, specify NOSSRANGE when you move the program to production.

Refer to your COBOL documentation for more information about these and other compiler options.

Compiler options that are not supported: The compiler option NAME is not supported.

Other ELAPREP parameters

In addition to the parameters that ELAPREP uses for COBOL compilation, ELAPREP supports other parameters to control the preparation process.

VMFMODE

The ELAPREP VMFMODE= parameter is used to specify the CMS file mode where the object that is being processed resides. It is also the file mode of the loadlib in which the created module is stored. It must be a single alphabetic character. The file mode must be accessed in R/W mode. The default file mode is A. The user-defined symbolic parameter VMFMODE can be used to specify the ELAPREP VMFMODE parameter.

VMLOADLIB

The ELAPREP VMLOADLIB parameter is used to specify the file name of a CMS loadlib in which the created module is stored. The default is EZEAPPL. The generator option /VMLOADLIB can be used to specify the ELAPREP VMLOADLIB parameter.

AMODE/RMODE

When ELAPREP uses the CMS LKED command to link-edit the generated program, it uses AMODE(31) and RMODE(ANY). You can specify AMODE=24, AMODE=ANY or RMODE=24 on the call to ELAPREP to override the defaults.

Use the following rules when generating or linking programs, tables, and map groups:

- Specify 24-bit addressing for the following:
 - Any program that calls a program linked AMODE(24)
 - The first program in the run unit if any generated program in the run unit is linked AMODE(24)
 - Tables and map groups if any program that uses the table is linked AMODE(24)

ERASETEXT

The ELAPREP ERASETEXT parameter is used to indicate whether ELAPREP should erase the text deck that was created by the COBOL compiler after ELAPREP uses it. The default is ERASETEXT=Y, which erases the text deck.

USERLKED

The USERLKED parameter is used to specify the name of user txtlibs that will be used during the link edit step of the ELAPREP. This is useful when your VisualAge Generator program needs to use static linkage for a non-VisualAge Generator object deck. More than one txtlib can be specified by separating the txtlib names with a /. The minidisks or SFS directories containing the txtlibs must be accessed before calling ELAPREP. By default, ELAPREP will not use any user txtlibs.

SQL preprocessing for VM

When a generated SQL program is prepared for running on VM, ELAPREP calls ELASQLAA EXEC to perform an SQL/DS VM preprocessor step. The preprocessor step creates a package (access module) that is stored in the SQL/DS VM database.

If you use ANSI SQL format (generation option /ANSISQL) for the SQL statements in the VisualAge Generator program, you may need to modify ELASQLAA to call your SQL preprocessor.

The SQL preprocessor step can be started in SQL/DS VM multiple program user mode or single program user mode. The default template for preparation runs in SQL/DS VM multiple program user mode. Multiple program user mode and single program user mode are used as follows:

Multiple program user mode

This is used if the relational database is shared among VM users, batch jobs, or both. The system operator must have started SQL/DS VM before the database can be used.

Single program user mode

This is used if the SQL/DS VM system and one or more programs run in the same virtual machine. In single program user mode, the run-time REXX exec starts SQL/DS VM and the SQL/DS VM preprocessor.

To change from multiple program user mode to single program user mode, use the SQLSTMDE symbolic parameter to alter the value.

You can customize the SQL/DS VM preprocessing. The project leader determines a set of standard SQL/DS VM preprocessor options that are needed for a project. Changes to options can be made in the default preprocessor options file for VM preparation, ELASQLPO PREPPP, which is stored on the VisualAge Generator Server for MVS, VSE, and VM disk. The preparation process can also be directed to use a different options file by setting the value of the symbolic parameter SQLPROPT to the new options file name. In VM, the SQL/DS VM preprocessor options file always has a file type of PREPPP.

If you specify single program user mode for preparing SQL/DS VM programs (user-defined symbolic parameter SQLSTMDE equals SINGLEUSER), then you can customize the ELASQLSO SQLPARM file, which is used to start the SQL/DS VM database. The ELASQLSO SQLPARM file is stored on the VisualAge Generator Server for MVS, VSE, and VM disk. If you want to specify a different SQL/DS VM startup options file, use the SQLSTOPT symbolic parameter. In VM, the SQL/DS VM startup options file always has a file type of SQLPARM.

The default package name for the SQL/DS VM program is the program name. If you want to specify a different package name, use the SQLPKGNM user-defined symbolic parameter.

In addition to SQLSTMDE, SQLPROPT, SQLSTOPT, and SQLPKGNM, other user-defined symbolic parameters that can be used to control the SQL/DS VM preprocessing step are SQLDBNAM and SQLUSRPW. See “Chapter 12. Symbolic parameters” on page 123 for more information.

Required SQL/DS VM options

The following options are required for SQL/DS VM usage. These options are included in the ELASQLPO PREPPP file, which is shipped with VisualAge Generator Server for MVS, VSE, and VM. Do not remove these options:

- COB2
- QUOTE

Setting additional SQL preprocessor options

Review the SQL preprocessor options in the SQL/DS VM program programming document for your version of SQL/DS VM. In addition, the following parameters are specified in the file ELASQLPO PREPPP, which is shipped with the VisualAge Generator Server for MVS, VSE, and VM and stored on the host services disk; these options can be modified or removed:

ISOL(CS)

Cursor stability isolation level

CTOKEN(YES)

Consistency token is stored in the expanded source and the package.

Setting COBOL run-time options for VM

You might need to set additional COBOL run-time options depending on the run-time library in use and the options your program requires.

COBOL run-time option

The WSCLEAR run-time option is required when using the COBOL run-time library. The WSCLEAR option is set during VisualAge Generator Server for MVS, VSE, and VM installation.

LE run-time options

LE run-time options for COBOL program programs are specified in the CSECT CEEUOPT. The LE product ships a sample file called CEEUOPT ASSEMBLE that can be used to override the LE product or installation defaults (CSECT CEEDOPT). VisualAge Generator Server for MVS, VSE, and VM ships a modified version of CEEUOPT that is called ELAEUOPT ASSEMBLE. It contains run-time options that are necessary for the VM environment. You can add additional LE run-time options to this file. If you do, you must assemble it and store the resulting text file on a file mode that is accessed when the preparation REXX exec is run. The file mode must be accessed before the VisualAge Generator Server for MVS, VSE, and VM disk because an ELAEUOPT TEXT file is located there.

The link-edit templates that are shipped with VisualAge Generator have a statement that includes ELAEUOPT. If ELAPREP determines that you are using COBOL, ELAPREP will include the ELAEUOPT TEXT file in the

generated program. If COBOL is being used, the INCLUDE ELAEUOPT statement in the link-edit file will have no effect because ELAPREP will set the FILEDEF for ELAEUOPT to DUMMY.

VisualAge Generator programs require that run-time option STORAGE(00,...) be used if the LE run-time library is utilized. ELAEUOPT ASSEMBLE specifies STORAGE=(00,NONE,NONE,8K). This overrides the LE product default of STORAGE=(NONE,NONE,NONE,8K).

For the DATA(31) COBOL option, specify the ANYWHERE suboption for the HEAP run-time option. Otherwise, all COBOL and VisualAge Generator Server for MVS, VSE, and VM storage is allocated with a 24-bit address. ELAEUOPT specifies HEAP=(32K,32K,ANYWHERE,KEEP,8K,4K). You may need to modify this based on your use of the DATA COBOL compiler option.

The TRAP and TERMTHDACT run-time options control error condition handling. TRAP specifies how LE routines handle abends and program interrupts. The LE documentation indicates that TRAP(ON) must be in effect in order for programs to run successfully. The ELAEUOPT ASSEMBLE file specifies TRAP=(ON) to ensure this setting.

TERMTHDACT controls the amount of information reported on a severe error detected by TERMTHDACT. TERMTHDACT(DUMP) produces an LE dump in addition to an LE message. ELAEUOPT ASSEMBLE file specifies TERMTHDACT=(DUMP).

Refer to the *COBOL Programming Guide* (SC26-4767) and the *LE Programming Reference* (SC26-3312) for more information on specifying LE run-time options.

Modifying LE user exits

In addition to setting appropriate LE run-time options, you can modify the LE assembler user exit (CEEEXITA) to request an ABEND instead of an error return for all abend codes from programs that can have updates to databases or for recoverable files that have not been committed. An ABEND forces a rollback; a return with an error code commits updates to the database.

The LE product ships a sample file called CEEEXITB ASSEMBLE that shows how to code a LE assembler user exit. VisualAge Generator Server for MVS, VSE, and VM ships a modified version of CEEEXITB that is called ELABXITB ASSEMBLE. It contains additional code to request an ABEND instead of an error return for all abend codes. You can add additional user exit processing to this file. If you do, you must assemble it and store the resulting text file on a file mode that is accessed when the preparation REXX exec is run. The file mode must be accessed before the VisualAge Generator Server for MVS, VSE, and VM disk because an ELABXITB TEXT file is located there.

The link-edit templates that are shipped with VisualAge Generator have a statement that includes ELABXITB. If ELAPREP determines that you are using COBOL, ELAPREP will include the ELABXITB TEXT file in the generated program. If COBOL is being used, the INCLUDE ELABXITB statement in the link-edit file will have no effect because ELAPREP will set the FILEDEF for ELABXITB to DUMMY.

Refer to the *LE Programming Guide* (SC26-4818) for more information on condition handling and the assembler user exit.

Examples of modifying templates

Examples showing how to modify templates and related procedures for specific tasks appear throughout the rest of this chapter. Before making any modifications to templates, copy all of the templates to a different directory and make your changes to these copies. You can edit template files using a standard editor.

When possible, samples of the modified templates have been included, with the modifications highlighted in **bold**. The detailed examples show how to modify the templates and procedures to accomplish the following tasks:

- “Adding a qualifier to the end user data set names”
- “Deleting COBOL source code from the workstation after preparation” on page 113
- “Deleting COBOL source on an MVS or VSE host” on page 114
- “Modifying a PSB name to match a batch program name” on page 115
- “Routing output to a system printer for an MVS/TSO CLIST” on page 116
- “Suppressing Personal Communications messages during file transfer” on page 117
- “Creating COBOL compile and link listings for CICS for OS/2” on page 118
- “Initializing the environment for CICS for OS/2” on page 119
- “Suppressing CICS translator, COBOL compile, and link messages for CICS for OS/2” on page 119

Adding a qualifier to the end user data set names

You can change the JCL to add qualifiers to the program user data set names. For example, you can add a code level or project identifier to the data set names where the output of the preparation job is sent. Symbolic parameters can represent these qualifiers in the JCL. To add an installation-defined symbolic parameter to represent a level qualifier in the program user data set names, create a symbolic parameter named LEVEL and add it to the following files:

- The preparation JCL templates on page 107

- The preparation procedures on page 108
- The run-time templates on page 111
- The preparation process template on page 113

You supply the value for the LEVEL symbolic parameter at generation time using the /SYMPARM generation option. The LEVEL symbolic parameter can represent test and production levels for a project, driver levels, or any other version of the product code that has meaning for your installation.

Adding a qualifier to a preparation template

Figure 27 shows how to add the LEVEL symbolic parameter to the preparation JCL templates. Make the change to copies of the preparation templates.

```
Change the string:
CGHLQ='%EZEPIID%'
to the string:
CGHLQ='%EZEPIID%',LEVEL='%LEVEL%'
```

Figure 27. Adding a qualifier to a preparation template

You can modify the EFK2MPCB template for preparing CICS for MVS/ESA programs without database access to add a LEVEL symbol.

Figure 28 shows the EFK2MPCB template after the LEVEL symbol is added. The modified lines shown in the following appear in **bold**.

```
//*****
//** EFK2MPCB - PREPARE MVSCICS PROGRAM WITH NO DB2 ACCESS
//**           CICS TRANSLATE, COMPILE AND LINK
//** MODIFIED PRH, 11/21/91, ADD LEVEL SYMBOL
//*****
//TCL EXEC ELATCL,MBR=%EZEMBR%,ENV=%EZEENV%,DATA=%EZEDATA%,
//           CGHLQ='%EZEPIID%',LEVEL='%LEVEL%'
//L.SYSIN DD *
  INCLUDE SELALMD(ELARSINC)
  INCLUDE SYSLIB(DFHEAT)
  NAME %EZEMBR%(R)
/*
```

Figure 28. EFK2MPCB template after the LEVEL symbol is added

Figure 29 on page 108 shows how the developer provides the value for the LEVEL symbol using the /SYMPARM generation option when generating a program that uses the modified template.

```
/SYMPARM=LEVEL, 'TEST'
```

Figure 29. Using a symbolic parameter to set the value for LEVEL during generation

Figure 30 shows the resulting preparation JCL statements.

```
//*****  
/** EFK2MPCB - PREPARE MVSCICS PROGRAM WITH NO DB2 ACCESS  
/** CICS TRANSLATE, COMPILE AND LINK  
/** MODIFIED PRH, 11/21/91, ADD LEVEL SYMBOL  
//*****  
//TCL EXEC ELATCL,MBR=MYAPPL,ENV=MVSCICS,DATA=31,  
// CGHLQ='PROJ100',LEVEL='TEST'  
//L.SYSIN DD *  
INCLUDE SELALMD(ELARSINC)  
INCLUDE SYSLIB(DFHEAI)  
NAME MYAPPL(R)  
/*
```

Figure 30. Resulting JCL statements

Adding a qualifier to a preparation procedure

Figure 31 shows how to add the LEVEL symbolic parameter to preparation procedures that currently have a CGHLQ symbol, by adding a default symbol definition to the PROC statement.

```
LEVEL='TEST',  
And change all occurrences of the following string:  
&CGHLQ..  
to:  
&CGHLQ..&LEVEL..
```

Figure 31. Adding a qualifier to a preparation procedure

The EFK2MPCB template uses procedure ELATCL. Figure 32 on page 109 shows the modification made to procedure ELATCL. The modified lines appear in **bold**.

```

//*****
//** ELATCL - CICS TRANSLATOR, COBOL COMPILE AND LINK-EDIT
//* MODIFIED, PRH, 11/21/91- ADDED LEVEL QUALIFIER
//*****
//*
//ELATCL PROC CGHLQ='USER',
//      LEVEL='TEST'
//      COBCICS='COB2.COB2CICS',
//      COBCOMP='COB2.COB2COMP',
//      COBLIB='COB2.COB2LIB',
//      ELA='ELA110',
//      DATA='31',
//      DFHLOAD='CICS211.LOADLIB',
//      ENV='IMSVS',
//      MBR=TEMPNAME,
//      RESLIB='IMSVS.RESLIB',
//      RGN=4096K,
//      SOUT='*',
//      SUFF='1$',
//      SP=,
//      DBCS=,
//      WSPC=500
//*
//*  PARAMETERS:
//*  CGHLQ   = COBOL GENERATION USER DATA SET HIGH LEVEL QUALIFIER
//*  LEVEL   = DRIVER LEVEL QUALIFIER
//*  COBCICS = COBOL CICS RUN TIME LIBRARY
//*  COBCOMP = COBOL COMPILER LIBRARY
//*  COBLIB  = COBOL RUN TIME LIBRARY
//*  DATA   = COMPILE OPTION FOR PLACING WORKING STORAGE
//*           ABOVE 16M LINE
//*  DFHLOAD = CICS LOAD LIBRARY
//*  ELA     = VisualAge Generator Server for MVS, VSE, and VM HIGH LEVEL QUALIFIER
//*  ENV     = COBOL GENERATION USER DATA SET ENVIRONMENT QUALIFIER
//*           (SHOULD BE EQUAL TO GENERATION TARGET ENVIRONMENT)
//*  MBR     = SOURCE NAME
//*  RESLIB  = IMS RESLIB LIBRARY
//*  RGN     = REGION SIZE
//*  SOUT    = SYSOUT ASSIGNMENT
//*  SUFF    = CICS PROGRAM NAME SUFFIX
//*  SP      = CICS/ESA SYSTEM PROGRAMMING TRANSLATOR OPTION
//*  DBCS    = DOUBLE BYTE CHARACTER SUPPORT
//*  WSPC    = PRIMARY AND SECONDARY SPACE ALLOCATION
//*

```

Figure 32. Changes to ELATCL (Part 1 of 2)

```

//*****
//*          CICS TRANSLATOR
//*****
//*
//T          EXEC PGM=DFHECP&SUFF,REGION=&RGN,
//          PARM=' COBOL2,QUOTE,NOSEQ&SP&DBCS '
//STEPLIB DD DISP=SHR,DSN=&DFHLOAD
//SYSPRINT DD SYSOUT=&SOUT,DCB=BLKSIZE=13300
//SYSPUNCH DD DISP=(MOD,PASS),DSN=&&SYSCIN,UNIT=VIO,
//          SPACE=(800,(&WSPC,&WSPC))
//SYSIN DD DISP=SHR,DSN=&CGHLQ..&LEVEL..&ENV..EZESRC(&MBR)
//*
//*****
//*          COMPILER THE COBOL PROGRAM
//*          IF THE RETURN CODE ON ALL PREVIOUS STEPS IS 4 OR LESS
//*****
//*
//C          EXEC PGM=IGYCRCTL,COND=(5,LT),REGION=&RGN,
//          PARM=(NOSEQ,QUOTE,OFFSET,LIB,RES,RENT,NODYNAM,OPT&DBCS,
//          'TRUNC(BIN)', 'NUMPROC(NOPFD)',NOCMPR2,'DATA(&DATA)')
//STEPLIB DD DISP=SHR,DSN=&COBCOMP
//SYSIN DD DISP=(OLD,DELETE),DSN=&&SYSCIN
//SYSLIB DD DISP=SHR,DSN=&ELA..SELACOPY
//SYSLIN DD DISP=(MOD,PASS),DSN=&&LOADSET,UNIT=VIO,
//          SPACE=(800,(&WSPC,&WSPC))
//SYSPRINT DD SYSOUT=&SOUT,DCB=BLKSIZE=13300
//SYSUDUMP DD SYSOUT=&SOUT,DCB=BLKSIZE=13300
//SYSUT1 DD SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=VIO
//SYSUT2 DD SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=VIO
//SYSUT3 DD SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=VIO
//SYSUT4 DD SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=VIO
//SYSUT5 DD SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=VIO
//SYSUT6 DD SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=VIO
//SYSUT7 DD SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=VIO
//*
//*****
//*          LINK-EDIT THE COBOL PROGRAM
//*          IF THE RETURN CODE ON ALL PREVIOUS STEPS IS 4 OR LESS
//*****
//*
//L          EXEC PGM=IEWL,COND=(5,LT),REGION=&RGN,
//          PARM='RENT,REUS,LIST,XREF,MAP,AMODE(31),RMODE(ANY) '
//SYSLIB DD DISP=SHR,DSN=&DFHLOAD
//          DD DISP=SHR,DSN=&COBCICS
//          DD DISP=SHR,DSN=&COBLIB
//          DD DISP=SHR,DSN=&RESLIB
//SELALMD DD DISP=SHR,DSN=&ELA..SELALMD
//SYSLIN DD DSN=&&LOADSET,DISP=(OLD,DELETE)
//          DD DDNAME=SYSIN
//SYSLMOD DD DISP=SHR,DSN=&CGHLQ..&LEVEL..&ENV..LOAD(&MBR)
//SYSPRINT DD SYSOUT=&SOUT,DCB=BLKSIZE=13300
//SYSUDUMP DD SYSOUT=&SOUT,DCB=BLKSIZE=13300
//SYSUT1 DD SPACE=(1024,(&WSPC,&WSPC)),UNIT=VIO
//*          PEND REMOVE * FOR USE AS INSTREAM PROCEDURE

```

Figure 32. Changes to ELATCL (Part 2 of 2)

Adding a qualifier to a run-time CLIST template

Figure 33 shows what changes to make to copies of the run-time CLIST templates to add the LEVEL symbolic parameter.

```
%EZEPIID%  
to the string:  
%EZEPIID%.%LEVEL%
```

Figure 33. Changes to add a qualifier to a run-time CLIST template change the string:

The EFK2METD template is used to generate a run-time CLIST for MVS/TSO main programs that do not use DL/I or DB2 databases.

Figure 34 on page 112 shows the modifications made to add a level qualifier to the data set name, which contains the executable load module in the EFK2METD template. The modified template is shown with changed lines appearing in **bold**.

```

/*****
/*  EFK2METD - EXECUTE TSO PROGRAM
/*      CALLED PROGRAM: %EZEMBR%
/*      TARGET ENVIRONMENT: %EZEENV%
/*      GENERATION DATE: %EZEGDATE%
/*      GENERATION TIME: %EZEGTIME%
/** MODIFIED PRH, 11/21/91, ADD LEVEL QUALIFIER
/*****
PROC 0 ALLOCATEONLY(NO)
CONTROL NOLIST NOFLUSH MSG
GLOBAL ELAAPPS
SET NAMESTART = &SYSINDEX(%EZEMBR%,&ELAAPPS) /*IS PRGM NAME IN STRING?*/
/* IF NAME FOUND, CLIST ALREADY RAN ONCE */
IF &NAMESTART > 0 THEN +
    EXIT /* EXIT IF ALREADY BEEN HERE */
ELSE +
    SET ELAAPPS = &ELAAPPS..%EZEMBR% /* ADD PROGRAM NAME TO */
/* LIST, SEPARATED BY PERIOD */
/*****
/*  ERROR EXIT
/*****
ERROR DO
    WRITE
    WRITE SYSPCMD = &SYSPCMD
    WRITE LASTCC = &LASTCC
    WRITE
    EXIT
END
IF &ALLOCATEONLY NE YES THEN DO /* EXIT IF ONLY ALLOCATION REQUEST */
    EXEC '%ELA,ELA110%.ELACLST(ELATALC)' +
        '&SYSUID..ELAPRINT ELAPRINT SYSDA FBA 0 1330 TRACKS 2 1 OLD'
    EXEC '%ELA,ELA110%.ELACLST(ELATALC)' +
        '&SYSUID..ELASNAP ELASNAP SYSDA VBA 0 4096 CYLINDERS 2 1 OLD'
    EXEC '%ELA,ELA110%.ELACLST(ELATALC)' +
        '&SYSUID..EZEPRINT EZEPRINT SYSDA VBA 0 4096 TRACKS 5 2 OLD'
    EXEC '%ELA,ELA110%.ELACLST(ELATALC)' +
        '&SYSUID..SYSABOUT SYSABOUT SYSDA FB 0 0 TRACKS 5 2 OLD'
    EXEC '%ELA,ELA110%.ELACLST(ELATALC)' +
        '&SYSUID..SYSOUT SYSOUT SYSDA FB 0 0 TRACKS 5 2 OLD'
END
/*****
/*  FILE ALLOCATIONS FOR THIS PROGRAM, TRANSFERRED-TO PROGRAMS
/*  AND CALLED PROGRAMS.
/*****
?ALLOC?
IF &ALLOCATEONLY EQ YES THEN EXIT /* EXIT IF ONLY ALLOCATION REQ.*/
CALL '%EZEPID%.%LEVEL%.%EZEENV%.LOAD(%EZEMBR%)'
EXIT

```

Figure 34. Adding a qualifier for a run-time CLIST template

Adding a qualifier to a preparation process template

This section shows how to add the LEVEL symbolic parameter to the preparation process templates.

The EFK20PPM.TPL template is the preparation control template for preparation for the MVS environments.

Figure 35 shows the modifications made to the EFK20PPM.TPL template. The modified template is shown with changed lines appearing in **bold**.

```
MVS_TEMP_UPLOAD='Y'  
MVS_TSO_ALLOC='N'  
MVS_RECLENG='80'  
MVS_BLKSIZE='6160'  
MVS_RECFCM='F'  
MVS_UNIT='CYLINDER'  
MVS_SPACE='1,1'  
PROJECTID='%ezepid%.%LEVEL%'  
MVS_TIMEOUT='1'
```

Figure 35. Modified template

Deleting COBOL source code from the workstation after preparation

Because you make many changes to your program by using VisualAge Generator Developer rather than by editing the generated COBOL code, and because storage is often a concern, you might want to delete the source code from the workstation after each compilation. The preparation templates include code to make it easy for you to delete source code after preparation is completed successfully. Figure 36 shows code that is included in the EFK2OPP4.TPL, EFK2OPPM.TPL, EFK2OPPV.TPL, EFK2OPPC.TPL, and EFK2OPPO.TPL templates.

```
DELETE_FILES='N'
```

Figure 36. REXX sample for deleting COBOL source code on the workstation

If you want to delete source code and intermediate files stored on the workstation after successful preparation of a part, make the following change:

1. If your target system is an MVS, VM, or VSE environment, edit a copy of the EFK2OPCA.TPL template.
If your target system is CICS for OS/2, edit a copy of the EFK2OPXP.TPL template. If your target system is OS/400, edit a copy of the EFK2OPP4.TPL template.
2. Change the value of DELETE_FILES from N to Y.

Deleting COBOL source on an MVS or VSE host

This section shows how to modify the templates to delete COBOL source from a MVS/TSO system after successfully compiling, or compiling and linking, the source. The PREPARE process consists of several different templates. Each of these templates must be edited to ensure that the COBOL code is deleted.

The following steps show how to edit one of the preparation templates. The same procedure is used for each of the templates:

1. Edit a copy of the EFK2MPTA.TPL template.

Note: This is the preparation template for compiling and linking an MVS/TSO program.

2. Figure 37 shows what to add after the /*.

```
//DELETE EXEC PGM=IDCAMS,COND=(5,LT)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE %EZEPIID%.%EZEENV%.EZESRC(%EZEMBR%)
```

Figure 37. REXX Sample for deleting COBOL source code on an MVS or VSE host

These JCL statements delete the COBOL source part when the previous steps, which perform the compile and link, complete with a return code between 0 and 4, indicating successful completion.

Figure 38 on page 115 shows the modifications made to the EFK2MPTA.TPL template. The modified template is shown with changed lines appearing in **bold**.

```

//*****
//** EFK2MPTA - PREPARE TSO PROGRAM WITH NO DB2 ACCESS
//**          COMPILE AND LINK
//*****
//CL  EXEC ELACL,MBR=%EZEMBR%,ENV=%EZEENV%,DATA=%EZEDATA%,
//          CGHLQ='%EZEPI%'
//L.SYSIN DD *
  CHANGE ELAAPPL(%EZEMBR%)
  INCLUDE SELALMD(ELARMAIN)
  INCLUDE SELALMD(ELARSINT)
  ENTRY %EZEENTRY%
  NAME %EZEMBR%(R)
/*
//DELETE EXEC PGM=IDCAMS,COND=(5,LT)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  DELETE %EZEPI%.%EZEENV%.EZESRC(%EZEMBR%)

```

Figure 38. Modifications to EFK2MPTA.TPL

Modifying a PSB name to match a batch program name

The default run-time JCL templates for IMS BMP and MVS batch DL/I programs set the PSB name to the name of the PSB part specified for the program. Similarly, the default run-time CLIST template for MVS/TSO DL/I programs sets the PSB name to the name of the PSB part specified for the program. If your naming conventions require the DL/I PSB name to be the same as the batch program name, you can modify the DL/I batch run-time templates and change string %EZEPSB% to the string %EZEMBR%.

Figure 39 on page 116 shows the required modifications to template EFK2MEBC with the changed lines appearing in **bold**.

```

//*****
//** EFK2MEBC - EXECUTE MVS BATCH PROGRAM WITH DLI ACCESS
//**
//**          PROGRAM: %EZEMBR%
//**          TARGET ENVIRONMENT: %EZEENV%
//**          GENERATION DATE: %EZEGDATE%
//**          GENERATION TIME: %EZEGTIME%
//** MODIFIED, PRH, 12/5/91 - SET DL/I PSB NAME TO PROGRAM NAME
//*****
//*
//%EZEMBR% EXEC DLIBATCH,DBRC=Y,
//  MBR=%EZEMBR%,PSB=%EZEMBR%,BKO=Y,IRLM=N
//G.STEPLIB DD
//          DD
//          DD DSN=%COB2LIB,COB2.COB2LIB%,DISP=SHR
//          DD DSN=%ELA,ELA110%.SELALMD,DISP=SHR
//          DD DSN=%EZEPID%.%EZEENV%.LOAD,DISP=SHR
//* DFSVSAMP IS REQUIRED IF VSAM DATABASE - REPLACE PART WITH ONE
//* THAT HAS VALID BUFFER POOL SIZES FOR YOUR PROGRAM.
//G.DFSVSAMP DD DSN=%ELA,ELA110%.ELASAMP(ELAVSAMP),DISP=SHR
//G.ELAPRINT DD SYSOUT=*,DCB=(RECFM=FBA,BLKSIZE=1330)
//G.ELASNAP DD SYSOUT=*,DCB=(RECFM=VBA,BLKSIZE=4096)
//G.EZEPRINT DD SYSOUT=*,DCB=(RECFM=VBA,BLKSIZE=4096)
//G.SYSABOUT DD SYSOUT=*
//G.SYSOUT DD SYSOUT=*
?DD?

```

Figure 39. Modifications to template EFK2MEBC

Routing output to a system printer for an MVS/TSO CLIST

The templates for run-time CLIST procedures route the printer output to data sets. Figure 34 on page 112 shows an example of the EFK2METD template with modifications. Figure 40 shows two statements from that template.

```

EXEC '%ELA,ELA110%.ELACLST(ELATALC)' +
    '&SYSUID..ELAPRINT ELAPRINT SYSDA FBA 0 1330 TRACKS 2 1 OLD'
EXEC '%ELA,ELA110%.ELACLST(ELATALC)' +
    '&SYSUID..EZEPRINT EZEPRINT SYSDA VBA 0 4096 TRACKS 5 2 OLD'

```

Figure 40. Lines from EFK2METD to allocate output data sets for EZEPRINT

These two statements use the ELATALC CLIST to allocate output data sets for ELAPRINT (the error information from VisualAge Generator Server for MVS, VSE, and VM) and for EZEPRINT (printer map output from the program).

Figure 41 on page 117 shows modifications to make to the template if you prefer that this information always be routed to the system printer rather than a data set.

```

/* ALLOCATE ELAPRINT */
ALLOC FI(ELAPRINT) SYSOUT(&SOUT) BLKSIZE(1330) LRECL(133) + RECFM(F,B,A) REUSE
/* ALLOCATE EZEPRINT */
ALLOC FI(EZEPRINT) SYSOUT(&SOUT) BLKSIZE(658) LRECL(654) + RECFM(V,B,A) REUSE

```

Figure 41. Routing output to a system printer

If the map group contains DBCS printer maps, BLKSIZE(658) and LRECL(654) are required for EZEPRINT. If the map group does not contain any DBCS printer maps, BLKSIZE(141) and LRECL(137) are required.

To set the output class for SOUT (for example, SOUT(A)) in the PROC statement of the CLIST template, you must add a default value.

You can make similar changes to the ELASNAP, SYSABOUT, and SYSOUT allocation commands and to the other CLIST templates. Use the allocation command changes for ELAPRINT as a guide for changes to the other templates.

Suppressing Personal Communications messages during file transfer

During the preparation process, generation output files are transferred automatically to the host when the target environment is MVS, VM, or VSE. The Personal Communications session issues information and error messages during this transfer. You can suppress these messages by editing the following preparation templates for the MVS, VM, and VSE environments if you are using a SNA transfer method:

Table 30. Templates used to transfer to target system

Template name	Target environment
EFK2OPPM	MVS
EFK2OPPV	VSE
EFK2OPPC	VM

The following steps show how to modify the template:

1. Edit a copy of the template for your environment.
2. Figure 42 on page 118 shows the lines to change.

```

QUIET = 'N'      /* Set session parm QUIET option      */
                /* specifies whether Personal      */
                /* Communication messages will be issued */
                /* during the Prepare process. The   */
                /* default is to issue these messages, */
                /* but can be changed to 'Y' suppress */
                /* display of these types of messages. */

```

Figure 42. Lines to change to suppress messages

3. Change the value of QUIET from 'N' to 'Y'.

Creating COBOL compile and link listings for CICS for OS/2

Preparation for CICS for OS/2 includes performing a COBOL compile and link. An abbreviated COBOL compile listing is produced by the preparation process. If you want to produce a complete COBOL compile listing, you can edit the CICS for OS/2 preparation template, EFK2OPXP.TPL. You can generate a complete listing by performing the following steps:

1. Edit a copy of the EFK2OPXP.TPL template.
2. Figure 43 shows a statement that generates the listing.

```

ezecobol_list = '' /* Specifies whether a complete COBOL compile listing */
                  /* is desired in the prgname.LST file.          */
                  /* This variable is initially set to NULL, which indicates */
                  /* that the abbreviated listing will be created. Set this */
                  /* variable to 'Y' to create the complete listing.      */

```

Figure 43. Creating the listing file

3. Change the value of ezecobol_list from '' to 'Y' to create the complete COBOL compile listing.

During CICS for OS/2 preparation, a link listing file is produced with the default name of *prgname*.MAP. You can change the name of the link listing file by editing the CICS for OS/2 preparation template, EFK2OPXP.TPL. Perform the following steps to change the link listing file name:

1. Edit a copy of the EFK2OPXP.TPL template.
2. Figure 44 shows a statement that sets the name of the link listing file.

```

ezelink_list = '' /* Link listing file - default of      */
                  /* applname.MAP in GENOUT path can be */
                  /* overridden. Change '' to 'example' */
                  /* to obtain example.MAP in GENOUT  */
                  /* path.                          */

```

Figure 44. Naming the link listing file

3. Change the value of `ezelink_list` from `"` to the name of the link listing file you want created.

Initializing the environment for CICS for OS/2

During CICS for OS/2 preparation, the functions performed include initializing the environment variables for VisualAge Generator Server, CICS for OS/2, and COBOL.

You might not want the environment variables initialized for you. You can modify the CICS for OS/2 preparation template, `EFK2OPXP.TPL`, to suppress this function. To suppress the initialization, perform the following steps:

1. Edit a copy of the `EFK2OPXP.TPL` template.
2. Figure 45 shows a statement that determines whether to initialize the environment.

```
call_elaenv      =  '1' /* 1 specifies that ELAENV.COMD is      */
                  /* be called to initialize Generator      */
                  /* Server, CICS OS/2, and COBOL          */
                  /* environment variables. Can be         */
                  /* changed to 0 to suppress the call.    */
```

Figure 45. Initialize the environment statement

3. Change the value of `call_elaenv` from `'1'` to `'0'`.

Suppressing CICS translator, COBOL compile, and link messages for CICS for OS/2

During CICS for OS/2 preparation, the preparation process issues messages. The CICS translator, COBOL compiler and linker can also issue messages. Sometimes you might not want to see those messages. You can modify the CICS for OS/2 preparation template, `EFK2OPXP.TPL`, to suppress messages from the CICS translator, COBOL compiler, and linker. To suppress the issuing of these messages, perform the following steps:

1. Edit a copy of the `EFK2OPXP.TPL` template.
2. Figure 46 shows a statement that suppress the messages.

```
ezemsg_mode     =  '' /* CICS translator, compile, and LINK */
                  /* informational message output mode */
                  /* The default, '', directs these */
                  /* types of messages to STDOUT. To */
                  /* suppress these types of messages */
                  /* change '' to '>>NUL'          */
```

Figure 46. Suppress CICS translator, COBOL compiler and linker messages

3. Change the value of `ezemsg_mode` from `"` to `'>>NUL'`.

Processing templates

This section shows an example of how a template is automatically processed. The sample template produces JCL when processed.

Template EFK2MPCB is selected as the template for automatically generating preparation JCL for a program that has only file access (no database access) and uses the CICS for MVS/ESA environment as the target environment.

This example section includes the following:

- The sample of the template before modification
- The values of the symbolic parameters for this example
- The resulting JCL

EFK2MPCB before modification

Figure 47 shows EFK2MPCB before modification.

```
//*****  
/** EFK2MPCB - PREPARE MVSCICS PROGRAM WITH NO DB2 ACCESS  
/**          CICS TRANSLATE, COMPILE AND LINK  
//*****  
//TCL EXEC ELATCL,MBR=%EZEMBR%,ENV=%EZEENV%,DATA=%EZEDATA%,  
//          CGHLQ='%EZEPIID%'  
//L.SYSIN DD *  
  INCLUDE SELALMD(ELARSINC)  
  INCLUDE SYSLIB(DFHEAI)  
  NAME %EZEMBR%(R)  
/*
```

Figure 47. EFK2MPCB before modification

The template contains symbolic parameters delimited by percent signs (%).

The values that are automatically substituted for the symbolic parameters are based upon the type of part being generated. See “Chapter 12. Symbolic parameters” on page 123 section for detailed information about symbolic parameters.

Values for the symbolic parameters

Table 31 shows the values of the symbolic parameters for this part as defined for this example.

Table 31. Values of the symbolic parameters for the example

Symbol	Value	Source of value
EZEMBR	MYAPPL	Program for which JCL is being generated
EZEENV	MVSCICS	Target environment

Table 31. Values of the symbolic parameters for the example (continued)

Symbol	Value	Source of value
EZEDATA	31	/DATA generation option
EZEPID	PROJ100	/PROJECTID (program user data set qualifier) generation option

The resulting JCL is used to prepare the generated part for run time. The JCL was created from the template and symbolic parameters. The generated JCL starts cataloged procedure ELATCL to prepare the generated program using the CICS translator, the COBOL compiler, and the linkage editor.

Preparation JCL created from the EFK2MPCB template

Figure 48 shows preparation JCL created from the EFK2MPCB template.

```
//*****
//** EFK2MPCB - PREPARE MVSCICS PROGRAM WITH NO DB2 ACCESS
//**          CICS TRANSLATE, COMPILE AND LINK
//*****
//TCL EXEC ELATCL,MBR=MYAPPL,ENV=MVSCICS,DATA=31,
//          CGHLQ='PROJ100'
//L.SYSIN DD *
  INCLUDE SELALMD(ELARSINC)
  INCLUDE SYSLIB(DFHEAI)
  NAME MYAPPL(R)
/*
```

Figure 48. Preparation JCL created from the EFK2MPCB Template

See “Modifying templates” on page 82 section for information on customizing a template.

Chapter 12. Symbolic parameters

Symbolic parameters are names that are substituted with values at generation time. Symbolic parameters can be used in the following places:

- Templates
- Specific generation and association options

The following generation and association options are valid for using symbolic parameters: /BIND, /GENOUT, /LINKAGE, /LINKEDIT, and /RESOURCE.

There are two types of symbolic parameters: predefined and user-defined.

Predefined symbols

The substitution values for predefined symbols are automatically set at generation. You do not define values for these symbols. The values for predefined symbols are derived from the following sources:

- Generation option values
- The type of program being generated
- Environment variable values
- The associated objects being generated
- The target environment
- System values
- File-related information

Predefined symbols start with the prefix EZE. An error is issued if a symbolic parameter beginning with the letters EZE is defined using the /SYMPARM generation option.

If a symbol is found in a template where it cannot be used, the symbol is deleted from the generated output line and a message is issued.

User-defined symbols

You can add your own symbolic parameters to templates. User-defined symbols enable you to specify substitution values at generation using the /SYMPARM generation option.

Part-related symbolic parameters

The following list contains the names and values of predefined symbolic parameters related to the part being generated.

EZECOBOLTYPE

The value is CALLEDAPPL, MAPGROUP, MAINAPPL, or TABLE, depending on the type of part being generated.

EZEDATA

The value 24 or 31 indicates the value of the DATA COBOL compile option. A value of 24 causes all storage to be placed below the 16MB boundary. A value of 31 enables storage above the 16MB boundary to be used. The value is set from the /DATA generation option. The maximum length for the symbolic parameter value is 2.

EZEDBCS

The value Y or N indicates whether double-byte characters are included. This value is set to Y if any DBCS characters are found in the part currently being generated. The maximum length for the symbolic parameter value is 1.

EZEDESTLIB

The value specifies the 1- to 10-character OS/400 library name where objects created during generation are placed when they are transferred by the preparation process.

EZEDESTNAME

The value specifies the 1- to 8-character Communication Manager (CM/2) PLU alias name for the AS/400 system you want to use when you transfer generated files to the host if the /PREP option is specified.

EZEDLI

The value Y or N indicates whether the program uses DL/I. This value is only set when you are generating programs. The maximum length for the symbolic parameter value is 1.

EZEENTRY

The value is the name of the primary entry point for the load module that is linked for the program being generated. For main programs in MVS/TSO, MVS batch, VM CMS, VM batch, and IMS BMP environments, the name is ELARMAIN. For all other programs, map groups, or tables, the name is the name of the program. EZEENTRY is set only when you are generating preparation JCL or the preparation REXX exec. The maximum length is 8.

Note: EZEENTRY cannot be used in the job statement.

EZEENV

The value identifies the target environment. The maximum length for the symbolic parameter value is 8.

EZEGBDATE

The value is the generation date in the format MM/DD/YY. The maximum length for the symbolic parameter value is 8.

EZEGBENOUT

The value is the path name of the directory where the generated part is placed. The value is set using the /GENOUT generation option. The maximum length for the symbolic parameter value varies depending on the actual directory names and the file system currently installed.

EZEGBMBR

The value is the name of the program, map group, or table specified in the GENERATE subcommand. The maximum length for the symbolic parameter value is 8.

If you have specified multiple parts on the GENERATE command, each part has a setting for EZEGBMBR.

EZEGBTIME

The value is the generation time in the format HH:MM:SS. The maximum length for the symbolic parameter value is 8.

EZEGBMBR

The value is the name of the part currently being generated. The value might be different from the symbolic parameter EZEGBMBR; EZEGBMBR always contains the name of the part specified on the GENERATE subcommand. The maximum length for the symbolic parameter value is 8.

EZEGBRPATH

The value is the path name of the directory where the generated part is placed. The value is set using the /GENOUT generation option when preparation is local and the target directory (/DESTDIR) of the preparation is remote. The maximum length for the symbolic parameter value varies depending on the actual directory names and the file system currently installed.

EZEGBMSG

The value is the identifier for the VisualAge Generator message table. The value is only set when generating programs. The maximum length for the symbolic parameter value is 4.

EZEGBID

The value is the data set high-level qualifier from the /PROJECTID generation option. The maximum length for the symbolic parameter value is 60.

EZEGBPREPDESTACCOUNT

The value is the account value required by TCP/IP when the remote

system requires an account value. This value is set using the /DESTACCOUNT generation option.

EZEPREPDESTHOST

The value is the name or numeric TCP/IP address of the target machine on which the target directory resides. This value is set using the /DESTHOST generation option.

EZEPREPDESTDIR

The value is the name of the directory on a remote machine that generated source code will be transferred to. This value is set using the /DESTDIR generation option.

EZEPREPDESTPASSWORD

The value is the password of the USERID that will be logged on to the target machine. This value is set using the /DETPASSWORD generation option.

EZEPREPDESTUID

The value is the password of the USERID that will be logged on to the target machine specified by the /DESTHOST generation option. This value is set using the /DESTUID generation option.

EZEPREPFTPMSBSCS

The value is set when you are transferring data that contains SBCS characters using FTP. This value sets the translation tables for the transfer of your preparation data. The default value that is run is ASCII. This value is set using the /FTPTRANSLATIONCMSBSCS generation option. By default, this option is not used by the generation process, because conversion automatically takes place locally when needed. If you want to use this option, you must edit EFKSND.REX and the preparation script.

EZEPREPFTPMDDBCS

The value is set when you are transferring data that contains DBCS characters using FTP. This value sets the translation tables for the transfer of your preparation data. This value is set using the /FTPTRANSLATIONCMDDBCS generation option. By default, this option is not used by the generation process, because conversion automatically takes place locally when needed. If you want to use this option, you must edit EFKSND.REX and the preparation script.

EZEPRESENDMDDBCS

The value is set when you are transferring data that contains DBCS characters using the SEND command. This value is set using the /SENDTRANSLATIONCMDDBCS generation option. By default, this option is not used by the generation process, because conversion automatically takes place locally when needed. If you want to use this option, you must edit EFKSND.REX and the preparation script.

EZEPRESESSION

The value is the session ID of the host emulator session you want to use when generated files are transferred to the host. This value is set using the /SESSION generation option.

EZEPREPSP

The value is set using the /SP generation option.

EZEPREPSQLDB

The value is the name of the database you want to use for SQL statement validation and for SQL preprocessing on CICS for OS/2 systems. This value is set using the /SQLDB generation option.

EZEPREPWORKDB

The value is the type of work database you want to use for the generated program. This value is set using the /WORKDB generation option.

EZEPSB

The value is the PSB name that is specified for the program being generated. The value is only set when generating programs. The maximum length for the symbolic parameter value is 8.

EZEPTYPE

The value is the part type value that is generated for generation output files. You can use the part type value to control the preparation steps when it is specified in the preparation templates.

When using VisualAge Generator only, you can use the part type value for the following:

- Store generation outputs in different directories when specified for the /GENOUT generation option
- Put the part type in the data set name when specified in a JCL template

Table 32 shows the valid part type values and their associated parts.

Table 32. Valid part type values and associated parts

Part type value	Associated parts
EZEBND	SQL BIND files
EZECLJ	Preparation CL job source
EZECLP	Preparation CL source
EZECLX	Run-time CLIST file
EZECPY	COBOL copybook
EZEDDS	Data description specifications
EZEDLL	DLL files

Table 32. Valid part type values and associated parts (continued)

Part type value	Associated parts
EZEFMT	Map group format module
EZEJCP	Preparation JCL file
EZEJCX	Run-time JCL file
EZELED	Link-edit control files
EZELNK	Link-edit control files
EZELST	Listing file
EZEMFS	MFS control blocks
EZEMGF	CL for message file preparation program
EZEPCT	CICS macro format transactions
EZEPPT	CICS macro format programs
EZEPRP	Preparation command file
EZERDO	CICS RDO format
EZERXP	REXX preparation command file
EZERXX	Sample REXX exec for run time
EZESRC	COBOL source files
EZETAB	Binary image tables

EZESQL

The value Y or N indicates whether SQL processes are used in the program. The value is only set when you are generating programs. The maximum length for the symbolic parameter value is 1.

EZETBLNAME

The value is the name of the message table that is generated into the OS/400 message file.

EZETPROC

The value is set to 8 characters in length where:

position 1-4

EZEUSRID (characters 1-4) is set from the **VAGen —Test General** Preferences tab or from machine name in C:\IBMLAN\IBMLAN.INI if the Preference is not set. If EZEUSRID is fewer than 4 characters, it will be padded with Zs.

position 5-8

MMSS (min,min,sec,sec) obtained from the system-generated

time For example, if EZEUSRID is set to USERID and the program is generated at 08:45:29, SYMPARM EZETPROC will be set to USER4529.

EZETRAN

The value is the transaction ID.

- For IMS programs, the value is the VisualAge Generator program name. The maximum length for the symbolic parameter value for IMS programs is 8.
- For CICS programs, the value is specified as the primary value for the /TRANSID generation option. If a primary value is not specified for the /TRANSID generation option, the first 4 characters of the VisualAge Generator program name are used for EZETRAN. The maximum length for the symbolic parameter value for CICS programs is 4.

EZETRANSFERTYPE

The value is the type of transfer used to transfer to the remote system. This value is set using the /TRANSFERTYPE generation option.

EZEUSRID

The value is the host user identifier, which is set from the **VAGen —Test General** Preferences tab. If EZEUSRID is not specified, the default is the computer name value, which is set in the C:\IBMLAN\IBMLAN.INI file.

EZEVMLOADLIB

The VM loadlib where the target load module is placed. This value is set by using the /VMLOADLIB generation option. The maximum length for the symbolic parameter value is 8.

EZEVELIB

The library where the target load module is placed. This value is set by using the /LIB generation option. The maximum length for the symbolic parameter value is 16.

EZEXAPP

The name of a program that receives control from the program that is being generated. Control is passed by using a CALL, DXFR, or XFER statement or the /RT generation option.

The VisualAge Generator Developer uses EZEXAPP to set the name of the transferred-to program in the following situations:

- When ?DD? occurs in the run-time JCL template, the template insert control statement EFK2MCAL is included one time for each transferred-to or called program.

- When ?DLBL? occurs in the run-time JCL template, the template insert control statement EFK2VCAL is included one time for each transferred-to or called program.
- When ?ALLOC? occurs in the run-time CLIST template, the template insert control statement EFK2MTCL is included once for each transferred-to or called program. When ?ALLOC? occurs in the run-time REXX exec template, the template insert control statement EFK2CVCL is included once for each transferred-to or called program.

Templates EFK2MCAL, EFK2VCAL, EFK2MTCL, and EFK2CVCL use EZEXAPP to refer to the name of the transferred-to or called program. Each time the templates are used, EZEXAPP is reset to the name of the current transferred-to or called program. The maximum length for the symbolic parameter value is 8.

Note: You can use EZEXAPP in templates EFK2MCAL, EFK2VCAL, and EFK2MTCL only.

File-related symbolic parameters

Symbolic parameters related to files are set only for templates that are used in the following instances:

- To generate DD or DLBL statements for program files in run-time JCL
- For ALLOC commands for program files in CLIST procedures
- For FILEDEF or DLBL commands in VM REXX execs

The following list describes file-related symbolic parameters and their values.

EZEBLK

The value is the file block size. For a fixed length record, it is set to the record length. For a variable length record, set to the record length plus 8 bytes. The maximum length for the symbolic parameter value is 5. Valid file types are SEQ, SEQRS, and GSAM.

EZEDBD

The value is the database name from the VisualAge Generator PSB. The maximum length for the symbolic parameter value is 8. This value is set for DL/I records.

EZEDD

The value is the DD name set from the record definition file name. The maximum length for the symbolic parameter value is 8. Valid file types are SEQ, SEQRS, GSAM, VSAM, and VSAMRS.

EZEDLBL

The value is the VSE DLBL or TLBL name set from the record

definition file name. The maximum length for the symbolic parameter value is 7. Valid file types are SEQ, VSAM, and VSAMRS.

EZEDSN

The value is the MVS data set, VSE library name set, or the VM CMS file name from the system resource name that is specified at generation. The maximum length for the symbolic parameter value is 56. Valid file types are SEQ, SEQRS, GSAM, VSAM, and VSAMRS.

EZELRECL

The value is the file logical record length. For a fixed length record, this symbol is set to the record length. For a variable length record, this symbol is set to the record length plus 4 bytes. The maximum length for the symbolic parameter value is 8. Valid file types are SEQ, SEQRS, and GSAM.

EZERECFM

The value is the file record format. For a fixed length record, this symbol is set to FB. For a variable length record, this symbol is set to VB. The maximum length for the symbolic parameter value is 2. Valid file types are SEQ, SEQRS, and GSAM.

User-defined symbolic parameters

The following is a list of user-defined symbols and their values. You can also define your own symbols.

COB2LIB

The value is the name for the LE run-time library used with COBOL. The default value is CEE.SCEERUN. The maximum length for the symbolic parameter value is 60.

COBLIST

The value indicates whether the COBOL compiler listing is printed. This is used for the VM environment only. Refer to the compiler documentation for your compiler for valid values for compile listings.

ELA

The value is the high-level qualifier of the product data sets for VisualAge Generator Server for MVS, VSE, and VM for MVS. The default value is ELA.V1R2M0. The maximum length for the symbolic parameter value is 60.

DBDLIB

The value is the name of the IMS/VS DBD library. The default value is IMS.DBDLIB. The maximum length for the symbolic parameter value is 60.

DSNLOAD

The value is the name of the DB2 load library. The default value is DSN.SDSNLOAD. The maximum length for the symbolic parameter value is 60.

DSYS

The value is the name of the DB2 subsystem. If an MVS system has more than one version of DB2 installed, the DSN system command uses this value to indicate which DB2 version to use for the current job. This value is used in the MVS batch preparation and run-time templates and the MVS/TSO run-time CLIST. The default value is DSN. The maximum length for the symbolic parameter value is 4 characters.

EZUAUTH

The value sets the value for the COBOL program AUTHOR statement. The maximum length for the symbolic parameter value is 60 characters.

EZUINST

The value sets the value for the COBOL program INSTALLATION statement. The maximum length for the symbolic parameter value is 60 characters.

PSBLIB

The value is the name of the IMS PSB library. The default value is IMS.PSBLIB. The maximum length for the symbolic parameter value is 60.

PROCLIB

The value is the name of the library where VisualAge Generator Server for MVS, VSE, and VM procedures for VSE are stored. The default value is PRD2.EZELIB. The maximum length for the symbolic parameter value is 16.

PWRCLASS

The value is the class that is used in all POWER job entry control language (JECL) statements. The default value is 0. The maximum length for the symbolic parameter value is 2.

RESLIB

The value is the name of the IMS RES library. The default value is IMS.RESLIB. The maximum length for the symbolic parameter value is 60.

SQLDBNAM

The value is the name of the DB2/VSE or SQL/DS VM database that is used during the preparation process. The default value for DB2/VSE is SQLDB_340. The default value for SQL/DS VM in single program user mode is SQLDBA. The default value for SQL/DS VM in

multiple program user mode is the database specified on the last SQLINIT EXEC processed on the virtual machine.

SQLPKGNM

The value is the name that is used as the DB2/VSE or SQL/DS VM package name. You can specify the name fully qualified with the owner name. If you do not specify the name fully qualified, the default value for the owner name is the user ID specified in the SQLUSRPW symbolic parameter. The default value is the current program name. The maximum length for the symbolic parameter value is 17.

SQLPROPT

The value is the name of the part that contains the DB2/VSE precompiler options or the name of the file containing the SQL/DS VM precompiler options. The VSE part can be in any library in the source chain. The VM CMS file must have a file type of PREPPP. In VM CMS, the first file found with the specified file name and a file type of PREPPP will be used. The default value for VSE is ELASQLPR. The default value for VM is ELASQLPO. The maximum length for the symbolic parameter value is 8.

SQLSTMDE

The value is the DB2/VSE or SQL/DS VM startup mode that is used by the preparation process. Valid values are MULTIUSER and SINGLEUSER. The default value is MULTIUSER.

SQLSTOPT

The value is the name of the DB2/VSE part or the file name of the SQL/DS VM file that contains the startup options that are used when starting DB2/VSE or SQL/DS VM in the SINGLEUSER mode. The part can be in any library in the source chain. In VM, the file must have a file type of SQLPARM; the first file found with the specified file name and a file type of SQLPARM will be used. The default value for VSE is ELASQLST. The default value for VM is ELASQLSO. The maximum length for the symbolic parameter value is 8.

SQLUSRPW

The value is the DB2/VSE or SQL/DS VM user ID and password. The following example shows the format of the password:

```
user ID/password
```

The default value for VSE is SQLDBA/SQLDBAPW. The default value for SQL/DS VM is the VM user ID for the SQL/DS VM user. The maximum length for the symbolic parameter value is 8 characters for the user ID and 8 characters for the password. These are separated by a slash (/), which results in a total maximum length of 17 for the symbolic parameter value.

VMFMODE

The value is the target VM file mode that will be used by the program generator when it transfers files to the host. This value is also set in the preparation REXX exec and is passed as a parameter to the ELAPREP exec. It is the file mode that tells ELAPREP where the input files for preparation exist and where to store the output of the preparation. The default value is 'A'. The maximum length of the symbolic parameter value is 1.

VMDISKADDR

The value is the target VM disk address used when transferring files to the host using FTP. The default value is 191.

VUSERLIB

The value is the name of an additional library that is used for link-editing elements for VSE programs with statically linked parts. This library can also be specified by adding it to the templates used for VSE or by making the library part of the permanent LIBDEF search chain. The default value is PRD2.EZELIB. The maximum length for the symbolic parameter value is 16.

Creating user-defined symbolic parameters

User-defined symbolic parameters can be from 1- to 8-alphanumeric characters and cannot begin with the letters EZE. You can define a new symbol and its value. You can also define values for symbols used in the default templates that are provided by the VisualAge Generator Developer. See “User-defined symbolic parameters” on page 131 for a list of the symbols used in the default templates.

Symbolic parameters are delimited by the percent sign (%). A default value can be defined in a template for an instance of a symbol by placing a comma and the default value after the symbol name.

Figure 49 shows an example of defining a default value for a symbolic parameter.

```
%MYSYMBOL%  
%MYSYMBOL,PROJECT1%
```

Figure 49. Defining a default value

The first line shows how to define a symbolic parameter with no default value. On the second line, the default value PROJECT1 is assigned to the user-defined symbol.

If a template that is used in generation contains a symbol that has not been assigned a value using the /SYMPARM generation option, the default value is

used. If a default value has not been defined, an informational message is issued and generation continues. No value is substituted for the symbol, and the symbol is removed from the generated output.

Note: If you define a default value for a symbolic parameter, the default value is used only for that occurrence of the symbolic parameter. If you use the same symbolic parameter again, the default value is not used. If you want to use a symbolic parameter multiple times with the same default value, you must specify the default value every time you want the default value to be used.

Assigning values to user-defined symbolic parameters

Substitution values for user-defined symbolic parameters are specified using the /SYMPARM generation option. Values can contain special characters. Values specified for the /SYMPARM generation option are enclosed in single quotation marks ('). To include a single quotation mark in the symbolic parameter value, the quotation mark must be specified twice.

Figure 50 shows an example of including single quotation marks in symbolic parameter values.

```
/SYMPARM=ELA, 'TEST' 'S'
```

Figure 50. Including single quotation marks

In this example, the value of the ELA symbolic parameter is set to TEST'S.

During generation, the templates are searched for symbol names that match the symbols defined using the /SYMPARM option. If a match is found, the name in the template is replaced with the value from the /SYMPARM option.

Chapter 13. Outputs of COBOL generation

The output of generation is COBOL source. Generation also creates related objects needed to prepare and run your program. These outputs contain the information necessary to transfer files and start the appropriate compile and link processes.

If you are generating a Web transaction program, output includes Java source and related objects. For more information on the generated files, refer to “Chapter 17. Outputs of Web transaction program generation” on page 183.

See “Chapter 14. Preparation process for COBOL generation” on page 153 for more information on preparing these outputs.

VisualAge Generator produces the following types of COBOL:

- COBOL for MVS, VM, and VSE environments
- ILE COBOL/400 for the OS/400 environment
- IBM VisualAge for COBOL for OS/2 for the CICS for OS/2 32-bit environment

The COBOL source code is compatible with the IBM COBOL compiler, the AD/Cycle COBOL compiler, and the LE run-time library. Generated program and map group objects are environment dependent. All program or map group objects are generated for one environment and cannot be used in another environment.

Tables can be generated as COBOL programs or as binary tables. Tables generated as COBOL programs for MVS, VM, and VSE are system dependent, but not environment dependent. A table generated and prepared for one environment can be used in another environment on the same system without generating the table for that environment again.

Tables generated for the CICS for OS/2 environment and the OS/400 environment are generated as binary tables. Binary tables are complete tables that do not require any preparation.

Generated objects are stored in the directory specified on the /GENOUT generation option. If /GENOUT is not specified, the generation objects are stored in the directory where the server process is running.

Objects generated for all part types

Table 33 shows the objects generated for program, map group, and table part types. Descriptions of these generated objects follow the table.

In all the environments listed, the preparation script is used in conjunction with preparation JCL, REXX, or CLJ to upload the files that need to be used in the COBOL compile step. The preparation JCL, REXX, and CLJ contain instructions to prepare the source for the load modules.

Table 33. Objects for all part types

Object type	File name	Run-time environments	Generation options controlling production	Modifiable after generation
Listing file	<i>partname.LST</i>	ALL	/LISTING /LISTINGONERROR	NO
Preparation script	<i>partname.PRP</i>	ALL	/PREPFILE	YES
Preparation JCL	<i>partname.JCP</i>	MVS and VSE environments	/PREPFILE	YES
Program and transaction definitions	<i>partname.PPT</i> <i>partname.PCT</i>	CICS for MVS/ESA CICS for VSE/ESA CICS for OS/2	/CICSENTRIES	Review and possible modification required
Table binary image	<i>tblname.TAB</i>	OS/400	/GENTABLES	NO
Data definition specification	<i>partname.DDS</i>	OS/400	/CREATEDDS	YES
Message file	<i>partname.MSG</i>	OS/400MSG	/GENTABLES	YES
Preparation CL	<i>partname.CLP</i>	OS/400CLP	/PREPFILE	YES
Preparation job stream	<i>partname.CLJ</i>	OS/400	/PREPFILE	YES
Message file job stream	<i>tblname.CLJ</i>	OS/400	/GENTABLES	YES
Preparation REXX	<i>partname.RXP</i>	CICS for OS/2 VM environments	/PREPFILE	YES

See “Chapter 29. Generation command and option descriptions” on page 297 for detailed information about options used on the subcommands.

Table 34 shows the generation outputs for the MVS environments.

Table 34. Generation outputs for MVS

Element	File name and extension	Uploaded	Environment
COBOL source	xxxxxxx.CBL ¹	Yes	All
Listing file	xxxxxxx.LST	No	All
Preparation script	xxxxxxx.PRP	No	All
Preparation REXX	xxxxxxx.RXP	No	All
Preparation JCL file	xxxxxxx.JCP	Yes	All
Sample run-time JCL	xxxxxxx.JCX	Yes	MVS batch, IMS BMP
Program definitions	xxxxxxx.PPT	Yes	CICS for MVS/ESA
DB2 binding statements	xxxxxxx.BDC	Yes	All
Sample run-time CLIST	xxxxxxx.CLX	Yes	MVS/TSO
Transaction definitions	xxxxxxx.PCT	Yes	CICS for MVS/ESA
Linkage control	xxxxxxx.LNK	Yes	All
Format objects	xxxxxxx.FMT	Yes	IMS/VS, CICS for MVS/ESA, TSO
MFS source	xxxxxxx.MFS	Yes	IMS/VS, IMS BMP
COBOL copybook	xxxxxxx.CPY	Yes	IMS/VS, IMS BMP

¹xxxxxxx indicates an 8-character file name.

Table 35 shows outputs for the VSE environments.

Table 35. Generation outputs for CICS for VSE/ESA and VSE batch

Element	File name and extension	Uploaded	Environment
COBOL source	xxxxxxx.CBL ¹	Yes	All
Listing file	xxxxxxx.LST	No	All
Preparation script	xxxxxxx.PRP	No	All
Preparation JCL file	xxxxxxx.JCP	Yes	All
Sample run-time JCL	xxxxxxx.JCX	Yes	VSE batch
Program definitions	xxxxxxx.PPT	Yes	CICS for VSE/ESA

Table 35. Generation outputs for CICS for VSE/ESA and VSE batch (continued)

Element	File name and extension	Uploaded	Environment
Transaction definitions	xxxxxxx.PCT	Yes	CICS for VSE/ESA
Linking files	xxxxxxx.LNK	Yes	All
Format objects	xxxxxxx.FMT	Yes	CICS for VSE/ESA

¹xxxxxxx indicates an 8-character file name.

Table 36 shows the generation outputs for the VM environments.

Table 36. Generation outputs for VM CMS and VM batch

Element	File name and extension	Uploaded	Environment
COBOL source	xxxxxxx.CBL ¹	Yes	All
Listing file	xxxxxxx.LST	No	All
Preparation script	xxxxxxx.PRP	No	All
Preparation REXX	xxxxxxx.RXP	Yes	All
Sample run-time REXX exec	xxxxxxx.RXX	Yes	All
Linking files	xxxxxxx.LNK	Yes	All
Format objects	xxxxxxx.FMT	Yes	VM CMS

¹xxxxxxx indicates an 8-character file name.

Table 37 shows the generation outputs for the CICS for OS/2 environments.

Table 37. Generation outputs for CICS for OS/2

Element	File name and extension
COBOL source for print map programs	xxxxxxx.CBL ¹
COBOL source for program programs	xxxxxxx.CCP
Listing file	xxxxxxx.LST
Preparation script	xxxxxxx.PRP
Preparation REXX	xxxxxxx.RXP
Program definitions	xxxxxxx.PPT
Transaction definitions	xxxxxxx.PCT
Table binary image	xxxxxxx.TAB

Table 37. Generation outputs for CICS for OS/2 (continued)

Element	File name and extension
Format objects	xxxxxxxx.OBJ

¹xxxxxxxx indicates an 8-character file name.

Table 38 shows the COBOL generation outputs for OS/400.

Table 38. Generation outputs for OS/400

Element	File name and extension	Uploaded
COBOL source	xxxxxxxx.CBL ¹	Yes
Server CL setup program	xxxxxxxx.CLR	Yes
Messages table	xxxxxxxx.MSG	Yes
Listing file	xxxxxxxx.LST	No
Preparation script	xxxxxxxx.PRP	No
Preparation job stream	xxxxxxxx.CLJ	Yes
Data definition specification	xxxxxxxx.DDS	Yes
Table binary image	xxxxxxxx.TAB	Yes
Map group binary image	xxxxxxxx.FMT	Yes

¹xxxxxxxx indicates an 8-character file name.

Listing file

If the /LISTING generation option is specified, a source listing of the part being generated and its associates, including resource associations, generation options, and linkage table entries used, is produced. This listing is also produced if the /LISTINGONERROR generation option is specified and errors occur during generation.

Preparation script

A preparation script (*partname.PRP*) is generated when the part is generated with the /PREPFILE generation option specified.

The preparation script produced by a GENERATE subcommand contains logic to prepare the generated parts. If the generated program includes tables or maps, the logic for preparing them is contained in the preparation script.

The preparation script is used to control preparation. The following preparation files are created for each target system for each environment:

Table 39. Preparation executable for each target system

Preparation executable	Target environment
MVS	JCL
VSE	JCL
VM	Rexx
CICS for OS/2	Rexx
OS/400	CL

One preparation script is created for each GENERATE subcommand issued, even if the GENERATE subcommand generates multiple parts.

EFK2OPCA is the preparation script template for all environments. This template is repeated in the preparation script for each part generated.

The following profile templates contain information on the following:

- Environment-specific information for the transfer
- How each part type is to be moved to the preparation system

Table 40. Templates used to transfer to target system

Template name	Target environment
EFK2OPP4	AS/400
EFK2OPPM	MVS
EFK2OPPV	VSE
EFK2OPPC	VM
EFK2OPPO	OS/2 CICS

See “Chapter 14. Preparation process for COBOL generation” on page 153 for more information on the preparation script file.

Preparation JCL for MVS and VSE environments

A preparation JCL file (*partname.JCP*) is generated when the /PREPFILE generation option is specified. VisualAge Generator produces JCL statements to start the appropriate precompile, translate, compile, bind, and linkage editor job for the generated programs. Table 41 shows the required steps included in the JCL depending upon the program type.

Table 41. Required steps in the JCL for MVS and VSE

Program type	Environment	Required step
SQL programs	MVS	DB2 precompile and bind steps
SQL programs	VSE	DB2/VSE precompile step

Table 41. Required steps in the JCL for MVS and VSE (continued)

Program type	Environment	Required step
CICS programs	CICS	CICS conversion and must link with CICS modules
DL/I programs	MVS	Must link with the appropriate DL/I modules
MFS control blocks	IMS	Require an MFS assemble and link
Map group format modules	VSE and MVS	Require a link step
COBOL programs	VSE and MVS	Must compile and link

Only one preparation JCL file is created for each GENERATE subcommand issued, even if the GENERATE subcommand generates multiple parts. The JCL is produced from templates that you can modify to meet your requirements for preparation.

See “Modifying templates” on page 82 for more information.

Preparation REXX for VM and CICS for OS/2 environments

A preparation REXX file (*partname.RXP*) is generated when the /PREPFILE generation option is specified for CICS for OS/2 and VM environments. VisualAge Generator produces a REXX file that contains commands to start the appropriate precompile, compile, and linkage editor steps for the generated programs. Table 42 shows the required steps included in the REXX exec, depending upon the program type.

Table 42. Required steps in the REXX File

Program type	Required step
SQL programs	Requires SQL precompile, COBOL compile, and link edit
Map group format modules	Requires a link step
COBOL programs	Must compile and link

Only one preparation REXX file is created for each GENERATE subcommand issued, even if the GENERATE subcommand generates multiple parts. The REXX file is produced from templates that you can modify to meet your requirements for preparation. See “Modifying preparation templates and EXECs for VM environments” on page 97 for more information.

Program and transaction definitions for MVS and VSE environments

Sample table entries are generated for CICS environments.

Sample program and transaction definitions for CICS systems are generated as specified in the /CICSENTRIES options. The definitions can be generated using macro level or RDO format.

Objects generated for programs

Table 43 shows the objects generated for programs. Descriptions of these objects follow the table.

Table 43. Objects generated for programs

Object type	File name	Run-time environments	Generation options controlling production	Modifiable after generation
COBOL program	<i>prgmname.CBL</i>	MVS VM VSE OS/400	None	NO
COBOL program	<i>prgmname.CCP</i> ¹	CICS for OS/2	None	NO
Sample run-time CLIST	<i>prgmname.CLX</i>	MVS/TSO	/RUNFILE	Review and possible modification required
Sample run-time CLIST	<i>prgmname.JCX</i>	IMS BMP MVS batch VSE batch	/RUNFILE	Review and possible modification required
BIND command file	<i>prgmname.BND</i>	IMS BMP IMS/VS CICS for MVS/ESA MVS/TSO MVS batch	None	Review and possible modification required
Sample Run-time REXX	<i>prgmname.RXX</i>	VM CMS VM batch CICS for OS/2	/RUNFILE	Review and possible modification required

¹The extension CCP indicates a COBOL file that requires CICS conversion.

See “Chapter 29. Generation command and option descriptions” on page 297 for information about generation options.

COBOL program

The COBOL program is generated for programs.

The generated program is a COBOL program that contains the following information:

- Program control logic
- Logic for program processes, statement groups, and I/O operations
- Data for both the program and program control

The program control logic performs the following functions for a program, as needed:

- Initialization
- Termination
- Error reporting
- Segmentation support, including environment saving and restoration
- Transfer of control

Sample run-time CLIST

Sample run-time CLISTs are generated when the /RUNFILE generation option is specified. VisualAge Generator produces a sample run-time CLIST for running a program in the MVS/TSO environment.

The CLIST is produced from templates that can be modified. See “Modifying templates” on page 82 for more information.

The generated CLIST might not be complete and must be reviewed and modified if necessary before being used. For example, the CLIST for the generated program does not contain file allocations for programs that are started using EZEAPP on the XFER or DXFR statements. Comments in the CLIST indicate where the allocation commands for these programs need to be added. To build the final CLIST needed to run a set of programs in a run unit, you must edit the CLIST for the first main program and include commands to run the run-time CLISTs for any programs transferred-to using EZEAPP on a DXFR or XFER statement.

Refer to the *VisualAge Generator Server Guide for MVS, VSE, and VM* for information on tailoring run-time CLISTS.

Sample run-time JCL

Sample run-time JCL is generated when the /RUNFILE generation option is specified.

VisualAge Generator produces sample JCL for running programs in the IMS BMP, MVS batch, and VSE batch environments. The JCL does not contain a JOB statement; each person using the JCL must provide a JOB statement.

The JCL is produced from templates that can be modified. See “Modifying templates” on page 82 for more information on modifying the sample templates.

The generated JCL might not be complete, and, if necessary, you must review and modify the JCL before using it. For example, the JCL for the generated program does not contain any DD or DLBL statements for data sets used by other programs that can be started by CALL, DXFR, or XFER statements. Comments in the JCL indicate where DD or DLBL statements for these programs need to be added. To build the final JCL needed to run a set of programs as a run unit, you must edit the JCL for the first main program and include the DD or DLBL statements for any program called using CALL, DXFR, or XFER.

Refer to the *VisualAge Generator Server Guide for MVS, VSE, and VM* for information on tailoring the run-time JCL.

Sample run-time REXX for VM and CICS for OS/2

VisualAge Generator produces a sample run-time REXX for running programs in the VM CMS, VM batch, and CICS for OS/2 environments when the /RUNFILE generation option is specified. The run-time REXX is produced (with a file type of EXECX) from templates that can be modified. The exec contains commands to set up the program environment and a section that contains calls to generate FILEDEF or DLBL commands. This section calls the ELACALLC exec to generate either a FILEDEF or DLBL command for every file identified in the resource association file for the program.

The generated run-time REXX might not be complete and must be reviewed, and modified if necessary, before you use it. A few examples of why you might modify the REXX include:

- The REXX for the generated program does not contain file allocations for programs that are started using EZEAPP on the XFER or DXFR statements. Comments in the REXX indicate where the invocations of ELACALLC EXEC or FILEDEF/DLBL commands need to be added.
- Non-VisualAge Generator programs called directly by name are treated as VisualAge Generator program calls. An invocation of a run-time REXX for the non-VisualAge Generator program (whether or not it exists) is generated in the run-time REXX for the VisualAge Generator program. You must either create the to handle the call for the non-VisualAge Generator program or delete the call.

Each run-time REXX file type must be renamed to EXEC. Before you invoke the main run-time REXX that is needed to run a set of programs as a run unit, review and tailor all run-time REXX for that run unit.

BIND command file

A BIND command file is generated for programs that use DB2. The default BIND command file is produced either from templates shipped with

VisualAge Generator or from a BIND control part specified during generation. The BIND command file is transferred to the host during the preparation process.

You might want to review and modify the contents of this file.

Objects generated for map groups

Table 44 shows the objects generated for map groups. Descriptions of the object types follow the table.

Table 44. Objects generated for map groups

Object type	File name	Run-time environments	Generation options controlling production	Modifiable after generation
Map group format module	<i>mapgname</i> FM.FMT Represents the map group name	IMS/VS CICS for MVS/ESA MVS/TSO OS/400 VM CMS CICS for VSE/ESA	/GENMAPS /GENHELPMAPS	NO
Map group format module	<i>mapgname</i> FM.OBJ	CICS for OS/2	/GENMAPS /GENHELPMAPS	NO
Batch print services COBOL program	<i>mapgname</i> P1.CBL	IMS BMP MVS batch VSE batch	/MSP /GENMAPS /GENHELPMAPS	NO
Batch print services COBOL program	<i>mapgname</i> P1.CBL	VM batch	/GENMAPS /GENHELPMAPS	NO
Online print services COBOL program	<i>mapgname</i> .CBL	CICS for MVS/ESA MVS/TSO OS/400 VM CMS CICS for VSE/ESA CICS for OS/2	/GENMAPS /GENHELPMAPS	NO
MFS print services COBOL program	<i>mapgname</i> .CBL	IMS/VS IMS BMP	/MSP /GENMAPS /GENHELPMAPS	NO

Table 44. Objects generated for map groups (continued)

Object type	File name	Run-time environments	Generation options controlling production	Modifiable after generation
MFS control blocks	<i>mapgname.MFS</i>	IMS/VS IMS BMP	/MSP /GENMAPS /GENHELPMAPS	NO
COBOL copybook for MFS MID/MOD layout	<i>mapgname.CPY</i>	IMS/VS IMS BMP	/MSP /GENMAPS /GENHELPMAPS	NO

See “Chapter 29. Generation command and option descriptions” on page 297 for information about generation options.

Map group format module

The map group format module is generated for map groups. The map group format module is a generated structure that describes the map layout for terminal maps in the map group. VisualAge Generator builds the structure as an object module of the proper type for the target environment. The map group format module is produced only if the map group contains terminal maps.

Batch print services COBOL program

The batch print services COBOL program is generated for map groups that contain print maps.

The batch print services program is a COBOL program that formats data for line printers and writes the data to either the printer output file (directly to the printer, a VSE/POWER LST queue member, or QSAM file) or to a generalized sequential access method (GSAM) file. The batch print services program also performs SET map CLEAR and EMPTY functions for print maps. This program is used with programs that run in the IMS BMP, MVS batch, VM batch or VSE batch environments.

In VSE batch, the printer output file can be either a sequential file associated with a system logical unit number or a VSE/POWER LST queue member. The batch print services COBOL program contains support for both types of files and can be shared by multiple programs. When the printer output file is shared among multiple programs, you must consider the following:

- If the map group is generated by itself, a default value is generated for the system logical unit number for sequential printer output.

- If the map group is generated with a program, the resource association file, if one exists, determines the value of the system logical unit number.
- The value of the system logical unit number at run time is determined by the way the map group was most recently generated.

If you specify the generation option /MSP=ALL for either the IMS BMP or the MVS batch environment, the batch print services COBOL program includes support for both sequential and GSAM files. If the batch print services COBOL program is used by more than one program, the resource association information for that program determines the type of support used.

Online print services COBOL program

The online print services COBOL program is generated for map groups that contain print maps. The online print services program is a COBOL program that performs printer I/O, SET map CLEAR and EMPTY functions for print maps in CICS, MVS/TSO, OS/400 and VM CMS environments.

MFS print services COBOL program

The message format service (MFS) print services COBOL program is generated for map groups that contain print maps. In IMS/VS and IMS BMP environments, MFS is used to build the terminal and printer data stream. The MFS print services program is a COBOL program that provides print support by building MFS message output descriptors (MODs). The program also supports SET map EMPTY and CLEAR functions for print maps.

MFS control blocks

The message format service (MFS) control blocks are generated for map groups for the IMS/VS and IMS BMP target environments using MFS utility control statements. The following control blocks are generated:

- Device output format and device input format (DOF/DIF). These control blocks describe the messages that MFS sends to or receives from the display device. One format (FMT) is generated for each map group. One device (DEV) statement is generated for each 3270-type device that has a screen size large enough to contain one of the display maps defined in the map group. A DEV statement is generated for each printer device that has a line-length long enough to support any of the printer maps defined in the map group.

If all maps contain double-byte character set (DBCS) or mixed fields, only DBCS devices are supported. Single-byte maps are supported on both non-DBCS and DBCS devices.

Individual map formats are described as device pages (DPAGE) within the format.

- Message input descriptor (MID). This control block describes how MFS formats the input message so that the generated program can process it. A

MID is generated for each map group. The input message associated with an individual map is defined as a logical page (LPAGE).

- Message output descriptor (MOD). This control block describes the output message that the generated program sends to MFS. A MOD is generated for each map group. The output message associated with an individual map is defined as a logical page.

Refer to the IMS documentation for your system for additional information about the MFS control blocks.

COBOL copybook for MFS MID/MOD layout

The COBOL copybook for message format service (MFS) message input descriptor/message output descriptor (MID/MOD) layout is generated for map groups that contain terminal maps.

For terminal maps, a COBOL copybook is generated that defines the layout of the MFS message input and message output descriptors. This copybook can be used by non-VisualAge Generator programs that perform deferred message switching to or from VisualAge Generator programs. One copybook is generated for each map group.

Objects generated for tables

Table 45 shows objects generated for tables. Descriptions of the object types follow the table.

Table 45. Objects for tables

Object type	File name	Run-time environments	Generation options controlling production	Modifiable after generation
Table COBOL program	<i>tblname.CBL</i> ¹	CICS for MVS/ESA MVS/TSO MVS batch IMS/VS IMS BMP VM CMS VM batch CICS for VSE/ESA VSE batch	/GENTABLES	NO
Binary table	<i>tblname.TAB</i>	CICS for OS/2 OS/400	/GENTABLES	NO

Notes:

¹*tblname* represents the table name.

See “Chapter 29. Generation command and option descriptions” on page 297 for information about generation options.

Table COBOL program

The table COBOL program is generated for tables in the following environments:

- IMS BMP
- IMS/VS
- MVS batch
- CICS for MVS/ESA
- MVS/TSO
- VM CMS
- VM batch
- VSE batch
- CICS for VSE/ESA

The table program is a COBOL program that contains the table contents defined in program working storage. This enables tables to be generated independently of programs when the contents of a table need to be changed.

Binary table

A binary table is generated for tables in the following environments:

- CICS for OS/2
- OS/400

The binary table is a file containing the contents of the table. No preparation is required for tables generated for the CICS for OS/2 environment.

Modifying generation output

Do not modify the programs generated by VisualAge Generator Developer.

If you need to report problems in the generated program to the IBM Support Center, you might be asked to provide the COBOL program source generated with the generation option /COMMENTLEVEL, along with any inputs used during the generation process. You also might be asked for the external source format export files for the VisualAge Generator parts that were used to generate the program.

If you need to extend the logic of the program with your own code, write the extension as a separate program that interfaces with the generated program through a CALL statement or a transfer statement.

Do not copy or use VisualAge Generator Server service calls in your own non-VisualAge Generator programs. The service calls are not programming

interfaces available to customers. They are intended only for use in programs generated by VisualAge Generator Developer.

Refer to the *VisualAge Generator Design Guide* for information about the services available for use in your non-VisualAge Generator programs.

Chapter 14. Preparation process for COBOL generation

When generation is completed, the generated outputs must be prepared for run time, just as you prepare programs you might write yourself.

Preparation takes place automatically unless you specify the `/NOPREP` option. If you specify the `/NOPREP` option, preparation takes place as a separate process. The `/PREP` option is the default.

The preparation process includes the following steps:

- Transferring parts to the target environment, if needed
- Unicode conversion, if needed
- Running precompilers, compilers, and linkers

When you specify the `/PREPFILE` generation option without the `/PREP` generation option, the `PREPARE` subcommand can be used to prepare the generation output.

The `PREPARE` subcommand can also be used to restart preparation if the preparation process is not successful in a manner that does not require the parts be generated again. The `PREPARE` subcommand is used for preparing programs generated using any VisualAge Generator Developer.

See “Inputs for COBOL generation” on page 51 for tables showing the inputs and outputs.

Preparing parts for MVS, VSE, VM, CICS for OS/2 systems

The preparation script (*partname*.PRP) controls the transfer and preparation on the target system. To create the preparation script, specify the `/PREPFILE` generation option. For MVS and VSE target systems, a preparation JCL file (*partname*.JCP) is also created. For VM and CICS for OS/2 target systems, a preparation REXX (*partname*.RXP) is created. For OS/400 target systems, a preparation CL file (*partname*.CL and *partname*.CLP) are created. These files are submitted to the server as batch jobs to compile them.

When you generate for MVS and VSE, the preparation script contains instructions to transfer the source files and preparation JCL file to the target system. If the generated source contains DBCS characters and the programmable workstation you are using is DBCS-enabled, options are set on the `SEND` command to correctly transfer these characters. The preparation command file contains logic to submit the preparation JCL file to the MVS internal reader or VSE/POWER queue to start preparation processing. The

preparation JCL file contains the preparation job stream. It also contains statements to reference the previously transferred source.

When you generate for CICS for OS/2 and VM, the preparation script contains instructions to transfer the source files and the preparation REXX to the target system. The preparation script contains logic to execute the preparation REXX to start preparation processing for the program.

When you generate for OS/400, the preparation script contains instructions to transfer the source files and the preparation .CL and .CLP files to the target system. The preparation script contains logic to execute the preparation files to start preparation processing for the program.

Note: Preparation of a DBCS program must be performed on a DBCS-enabled programmable workstation.

For CICS for MVS/ESA and CICS for VSE/ESA environments using VisualAge Generator, the table entries are transferred to the host by the preparation process. The CICS tables on the host are not automatically updated with the generated definitions. The generated definitions might need to be modified before being used. The actual transaction name must be substituted in place of the symbol in the template. You are responsible for adding the definitions to the appropriate CICS tables on the host systems where the program is run.

Preparation script file contents

You can use the templates shipped with VisualAge Generator or you can create your own script file. See “Preparation script templates” on page 58 for information on the contents of the templates.

The preparation script file is divided into four sections. Each section is preceded by a tag. The tags are :PROFILE, :TYPE, :MESSAGES, and :CONTROL. Each section contains keywords that can be assigned values. The keywords are immediately followed by an equals sign (=) and the value associated with the keyword.

Preparation script file restrictions

- An asterisk (*) in column 1 indicates a comment line.
- A colon (:) in column 1 indicates a tag.
- The assignment operator is the equals sign. (for example, TRANSFER_TYPE='SNA').
- A space cannot precede or follow an assignment operator.
- The assigned values must be enclosed between two single quotation marks (').
- A line can contain multiple assignments, but they must be separated by a space (for example, SYSTEM='OS400' GENOUT='X:\GENOUT').

Preparation script sections and keywords

Table 46 describes each section and their keywords. An invalid keyword in a preparation file will cause the preparation to fail.

Table 46. Preparation script file contents

Section	Keyword	Default	Environment	Description
:PROFILE			All	Sets the general parameters for the transfer program
	TRANSFER_TYPE (required)		All	What protocol to use to transfer generated files to host. SNA is valid only for transfer to MVS, VM, and VSE. TCPIP is valid for all. LOCAL is valid only for OS/2.
	GENOUT (required)		All	Path where generated files reside
	SYSTEM (required)		All	Host generation environment
	PREP_SYSTEM (required)		All	The remote environment (OS400, OS2, VM, VSE, or MVS), which indicates to the script program which routine to run. For more details, see "Customizing the preparation process" on page 159.
	CONVERSION_TABLE		All	Conversion table to be used for the remote machine
	ENTRY_POINT	1	All	A version of the routine. This is used to indicate to the script program which procedure to run. For more details, see "Customizing the preparation process" on page 159.
	DELETE_FILES	N	All	Whether source and intermediate files should be deleted if preparation was successful. Deleting source files will require you to regenerate to run the prepare process again.
	PREP	D	All	Where a preparation exec will be submitted. The default value 'D' will submit the preparation exec using the host default (foreground for VM and background for all other environments).
	SESSION		All	SNA ONLY - Session that is connected to the host

Table 46. Preparation script file contents (continued)

Section	Keyword	Default	Environment	Description
	SENDFILE_OPTIONS	null string ('')	All	SNA ONLY. Additional options for sending files. For more information, see the documentation for the SEND command.
	QUIET	N	All	Whether Personal Communication messages should be issued during preparation execution. This can be particularly helpful when debugging upload failures.
	DESPASSWORD		All	Password for host. A password is needed when transferring files via TCPIP
	DESTACCOUNT		All	Account on destination host
	DESTUID		All	Host user ID
	DESTHOST		All	Host to connect to when using TCPIP
	OS2_DESTDIR		OS/2	Destination directory
	OS400_DESTLIB		OS/400	OS400 destination library. If no library is specified in EZEDESTLIB, QGPL is used, if specified in the template.
	VM_FMODE		VM (SNA)	The default VM file mode used for uploads to a VM session.
	VM_DISK_ADDR		VM (TCPIP)	The default VM disk address used for uploads to a VM session
	MVS_ALLOC_OPTIONS		MVS	The following information: <ul style="list-style-type: none"> • Record length for temporary file • Block size for temporary file • Record format for temporary file • Unit for allocation of the temporary file used for EZESRC. • Primary and secondary space quantities for the temporary file used for EZESRC.
	MVS_PROJECTID		MVS	Project identifier
	VSE_LIB		VSE	VSE library name
:TYPE			All	Sets the parameters for each file type

Table 46. Preparation script file contents (continued)

Section	Keyword	Default	Environment	Description
	PART_TYPE (required)		All	File type being processed. Valid values are: EZEJCP, EZEJCX, EZECLX, EZESRC, EZEPT, EZEPCF, EZEMFS, EZEFT, EZEFTS, EZECLR, EZECLJ, EZEMGF, EZEDDS, EZETAB
	WORKSTATION_EXT (required)		All	File extension on the workstation
	WORKSTATION_EXT_ALT		All	Alternate file extension on the workstation
	IS_BINARY	N	All	Whether the type of file being processed is binary
	PREP_SUBMIT	N	All	Whether the job should be submitted after preparation is complete
	SENDFILE_OPTIONS	Null string ("")	All	Options to be appended to the options specified by SENDFILE_OPTIONS in the :PROFILE section
	CNV_RECORD_LENGTH	80	All	The resulting record length per line after conversion. Use 0 to indicate variable length records, where end of record is denoted by a carriage return.
	USE_CSO_CONVERSION	Y	All	Whether a text file should be converted by the conversion program.
	OS400_FILENAME		OS/400	File name on OS/400
	OS400_FILENAME_SUFFIX		OS/400	Suffix to append to the OS/400 file
	MVS_EXT		MVS	File extension on MVS
	MVS_ALLOC_OPTIONS		MVS	Options that overwrite the options specified by MVS_ALLOC_OPTIONS in the :PROFILE section
	VM_FTYPE		VM	File type on VM
	VM_FTYPE_SQL		VM	Whether the file contains SQL
	VM_RECFCM		VM	Record format on VM
	VM_LRECL		VM	Record length
	VSE_FTYPE		VSE	File type on VSE

Table 46. Preparation script file contents (continued)

Section	Keyword	Default	Environment	Description
:MESSAGES	MSGCGxxx		All	Sets the messages issued by the transfer program
:CONTROL			All	The information about the generation outputs to be sent
	HAS_DBCS	N	All	Whether file contains DBCS
	NAME (required)		All	Name of file
	TYPE (required)		All	File type
	HAS_SQL	N	All	Indicates whether file contains SQL
	USE_EXT_ALT	N	All	Whether the alternate workstation file extension is used

The options MVS_TEMP_UPLOAD and MVS_TSO_ALLOC in previous release are replaced by the option ENTRY_POINT. MVS_TEMP_UPLOAD specifies whether or not source and other files uploaded during the preparation phase are initially loaded into a temporary file. By first uploading to a temporary file, you can avoid contention problems associated with use of MVS partitioned data sets. MVS_TSO_ALLOC is used only if MVS_TEMP_UPLOAD is set to Y. This specifies whether the file is to be created by EHLAPPI('N') or by a modifiable TSO allocated command. Note that this applies only to SNA transfer.

Use Table 47 to determine which ENTRY_POINT value to use in the preparation script.

Table 47. Preparation script ENTRY_POINT values

MVS_TEMP_UPLOAD	MVS_TSO_ALLOC	
	N	Y
N	2	(invalid combination)
Y	1	3

The option MVS_TIMEOUT is no longer used. If the system you are running is extremely slow, or if you are transferring large files, you can modify the value of pref.MVS_setPauseLen and pref.MVS_setLoopAmt within the efksnd.rex script. Setting a higher value of pref.MVS_setPauseLen increases the amount of time before the program checks the system again for ready. Setting a higher pref.MVS_setLoopAmt increases the number of times that the program checks the system for ready.

Setting PROFILE keywords on the command line

You can use the command line to set the profile key words. The keywords set on the command line override the values in the preparation script file. The syntax is as follows:

```
efkprep xxxx.prp /DESTUID='myuserid' /DESTPASSWORD='secret' /GENOUT='X:\MYDIR\GENDIR'
```

Customizing the preparation process

To customize your preparation process, you can modify the file efksnd.rex directly. Efksnd.rex is a rexx program that takes the generated preparation script as input. According to the assigned values retrieved, it first determines which routine to call. Then from within that routine, it issues file transfer commands to the remote system as well as issuing a preparation submit. Each routine is separated according to the remote system as well as the transfer type. Different versions of the routine are indicated by the last value at the end of the routine name.

The routine name is composed as follows:

```
prep_system.transfer_type  
.entry_point
```

where *prep_system* is the value assigned to the keyword PREP_SYSTEM, and similarly, for *transfer_type* and *entry_point*. For example, using version 2 of the routine for transferring to an MVS system using SNA transfer protocol will result in routine MVS.SNA.2 being evoked. We suggest that you add another version of the routine into efksnd.rex customization.

To customize the preparation process, take the following steps:

1. Make an backup copy of efksnd.rex.
2. Make an copy of the routine closest to what you need within efksnd.rex. You can put this routine at the end of the file or after the EXIT keyword within the file.
3. Create the proper name for this routine by using the naming convention described above. Choose a version value that has not been used. This will be the value that is entered as the ENTRY_POINT in the preparation template.
4. Modify and test the routine as needed.
5. Modify the appropriate preparation template so that the generated preparation script uses the newly added ENTRY_POINT value. For more information on template customization, see “Chapter 11. Templates for COBOL generation” on page 57.
6. Use the modified template to generate from VisualAge Generator.

Additional preparation steps

Additional preparation steps might be required before a program can be used. For example, you might need to do some of the following steps:

- Verify that the preparation steps completed successfully
- Modify run-time JCL, CLISTs, or the Run-time REXX
- Define transactions and programs to IMS or CICS

Refer to the *VisualAge Generator Server Guide for MVS, VSE, and VM* or *VisualAge Generator Server Guide for Workstation Platforms* for more information on preparation processing.

Getting ready for MVS preparation

Before running a preparation job stream on an MVS system, you must ensure that the data sets you need have been allocated. A CLIST file, named ELA110.ELACLST(ELACUSER), is provided by VisualAge Generator Server for MVS, VSE, and VM to do the allocation. The ELACUSER CLIST file allocates data sets for each user ID it is run under.

Figure 51 shows the naming conventions for the allocated data sets.

```
'&HLQ..&GENV..TYPE'
```

Figure 51. Naming conventions

The values are substituted as follows:

HLQ The high-level qualifier. This value defaults to the user ID.

GENV

The generated target environment. Valid values follow:

- MVSCICS
- TSO
- MVS BATCH
- IMSVS
- IMSBMP

TYPE The value for TYPE can be any of the following:

EZEBIND	BIND command file
EZECLST	Run-time CLIST file
EZECOPY	COBOL copybooks
EZEFOBJ	Map group format module
EZEJCLP	Preparation JCL
EZEJCLX	Run-time JCL
EZELINK	Link-edit control file

EZEMFS	MFS control blocks
EZEPCT	CICS program control table
EZEPPT	CICS program properties table
EZESRC	COBOL source code

You can also allocate data sets to a specific project by passing a `MVS_PROJECTID` value to the ELACUSER CLIST file. Refer to the *VisualAge Generator Server Guide for MVS, VSE, and VM* or the prologue of the ELACUSER CLIST file for more information on allocating data sets and how to pass a `MVS_PROJECTID` value to the ELACUSER CLIST file.

Transferring files to MVS systems

Preparation processes for MVS environments include transferring the files associated with the following:

- DB2 BIND command file (all MVS environments with DB2)
- Link-edit control statements (all MVS environments static link)
- COBOL copybook for MFS MID/MOD layout (IMS/VS and IMS BMP)
- Preparation JCL (all MVS environments)
- Sample run-time JCL (MVS batch and IMS BMP)
- Generated message format service (MFS) control block (IMS/VS and IMS BMP)
- COBOL source (all MVS environments)
- Map group format module (CICS for MVS/ESA, MVS/TSO, and IMS/VS)
- Sample run-time CLIST (MVS/TSO)
- CICS program definitions (CICS for MVS/ESA)

Getting ready for VSE preparation

The sublibrary `PRD2.EZELIB` is defined during installation of VisualAge Generator Server for MVS, VSE, and VM. This is the default library for preparation on VSE. You can specify a different library using the `/LIB` generation option.

Transferring files to VSE systems

Preparation processes for VSE environments include transferring the files associated with the following:

- Preparation JCL (all VSE environments)
- Sample run-time JCL (VSE batch)
- COBOL source (all VSE environments)
- Map group format module (all VSE environments)
- CICS program definitions (CICS for VSE/ESA)

Preparing parts for OS/400

The preparation script (*partname*.PRP) produces output on the target system for the generated parts by the preparation process. To create the preparation

script, specify the /PREPFILE generation option. For the OS/400 target system, a preparation CLJ file is also created.

Note: You must prepare a DBCS program on a DBCS-enabled programmable workstation.

Additional preparation steps

Verify that the preparation steps completed successfully by examining the compile listing and job log listing in the output queue on the OS/400 system. The output queue name is specified by the //BCHJOB command during preparation or by the job description used to define the batch job.

Refer to the *VisualAge Generator Server Guide for AS/400* document for more information about additional preparation steps.

Getting ready for OS/400 preparation

The default library for preparation is QGPL. You can specify a different library using the /DESTLIB generation option. Before you can use a library specified by the /DESTLIB generation option, you must copy the QVGN* files from QGPL to this library.

Transferring files to OS/400

Preparation processes for OS/400 environments include transferring the following files:

- Preparation CL
- Run-time CL
- Job stream CL
- COBOL source
- Map group format module
- Message CL
- Data Description Specifications (DDS)
- Tables

Preparing parts for CICS for OS/2

The preparation script (*partname.PRP*) produces output on the target system for the generated parts during the preparation process. To create the preparation script, specify the /PREPFILE generation option. For CICS for OS/2 target systems, a preparation REXX (*partname.RXP*) is created.

The preparation script contains all of the instructions necessary to prepare the generated parts. If a step in the preparation command file is not needed for a specific part, the step is ignored.

If you are developing on Windows NT and your target system is OS/2 or if you are developing on OS/2 and your target system is another OS/2 machine, the preparation REXX is executed on the target machine. If you are developing on your target machine, the REXX is executed locally.

Additional preparation steps

Additional preparation steps, such as the following, might be required before a program can be used:

- Verify that the preparation steps completed successfully.
- Define transactions and programs to CICS.

Outputs of CICS for OS/2 preparation

The outputs of CICS for OS/2 preparation are placed in the directory specified using the /GENOUT generation option.

The following files are produced during the preparation process:

- The .CBL file contains the translated COBOL source that was output from the CICS for OS/2 translator
- For DB2 programs generated for VisualAge for COBOL for OS/2, the .SQB file contains the translated COBOL source that was output from the CICS for OS/2 translator. This file is input to the DB2/2 Preprocessor, which produces the .CBL file and the .BND file.
- The .TRL file contains CICS translator messages
- The .OBJ file contains output from the COBOL compile
- The .DEF file contains the input to the link step
- The .MAP file contains the output from the link step

These files are not necessary for running a program and can be deleted automatically by modifying the EFK2OPXP template. See “Deleting COBOL source code from the workstation after preparation” on page 113 for information on modifying the template to delete the COBOL source.

The final outputs of the preparation process are the following:

- The .DLL file or files containing the executable program file
- The .BND file containing the SQL BIND commands
- The modified .PCF file, containing information on all parts generated and prepared for the program

Note: Abbreviated listing files are produced by default. If you want a complete listing file, you can modify the EFK2OPXP template. See “Creating COBOL compile and link listings for CICS for OS/2” on page 118 for information on modifying the template.

CICS for OS/2 table entries

Automatic update of program and transaction tables is not available for the CICS for OS/2 environment. Only sample entries are generated for the processing program table (.PPT) and program control table (.PCT). The program user must update the CICS definitions before using the generated program.

The following additions must be made to the table entries:

- A program control table (PCT) entry for each program that is to be run as a main transaction
 - A processing program table (PPT) entry for each online print mapping services program. Define the mapping services program to be a temporary resident program
 - A file control table (FCT) entry for each VSAM file
 - A destination control table (DCT) entry for each file assigned to a transient data queue
 - A PPT entry for each program distributed using CICS for OS/2 CAIM and CAEX transactions
 - Additional table entries are required for programs that call remote programs
- Refer to the CICS for OS/2 documentation for more information.

Analyzing preparation errors

The HPTCMD PREPARE subcommand issues return codes and messages to indicate the success of preparation.

Analyzing return codes

The return codes for preparation are as follows:

- | | |
|----|----------------------------------------------------------------------------|
| 0 | The preparation process was successful. |
| 4 | The preparation was successful, but some information messages were issued. |
| 8 | The preparation was not successful. |
| 12 | The command syntax is not valid. |
| 16 | The NLS identifier is not valid. |

Locating preparation error messages

If preparation was automatically started by specifying the /PREP option, preparation messages go to the same destination as messages returned by the GENERATE subcommand.

If you use the PREPARE subcommand, messages are written to the STDOUT destination for OS/2 commands. The default STDOUT destination is the current OS/2 session.

To route the messages to a file instead of displaying them in the current OS/2 session, add the following to the end of the PREPARE subcommand:

```
> filename 2>&1;
```

where *filename* is the file to write the preparation messages to. This is the standard OS/2 technique for routing command messages to a file.

Analyzing messages

If the preparation was not successful, do the following:

1. Review the messages to determine the cause of the failure.
2. Correct the problem.
3. If the messages indicate a problem in the generated parts, run the generation process again. You can automatically start the preparation process from generation by using the /PREP option.
4. If the problem was in the preparation process, you can start the preparation process again using the PREPARE subcommand.

Refer to the *VisualAge Generator Messages and Problem Determination Guide* for more information about the messages returned by the PREPARE subcommand.

Chapter 15. Command interface for COBOL generation

You can issue the VisualAge Generator Developer subcommands from a system prompt or from within a command file. The command HPTCMD implements the command interface. Subcommands are specified with the HPTCMD command and are followed by any required keywords and options. Comments can be imbedded in the commands. Comments begin with the characters `/*` and end with the characters `*/`.

Command processing can be started explicitly by issuing the START subcommand or implicitly by issuing any other VisualAge Generator Developer subcommand. The command continues running until it is ended, either by issuing the STOP subcommand or by closing the Generation Monitor window.

Starting the command opens a Generation Monitor window. The Generation Monitor window displays the command currently being processed and provides information showing what stage of generation the process has reached. You can cancel the currently processing command from the Generation Monitor window. Closing the Generation Monitor window ends any command currently being processed.

If you are generating programs using a generation server, refer to the *VisualAge Generator System Development Guide* for information about starting and stopping the Generation Monitor.

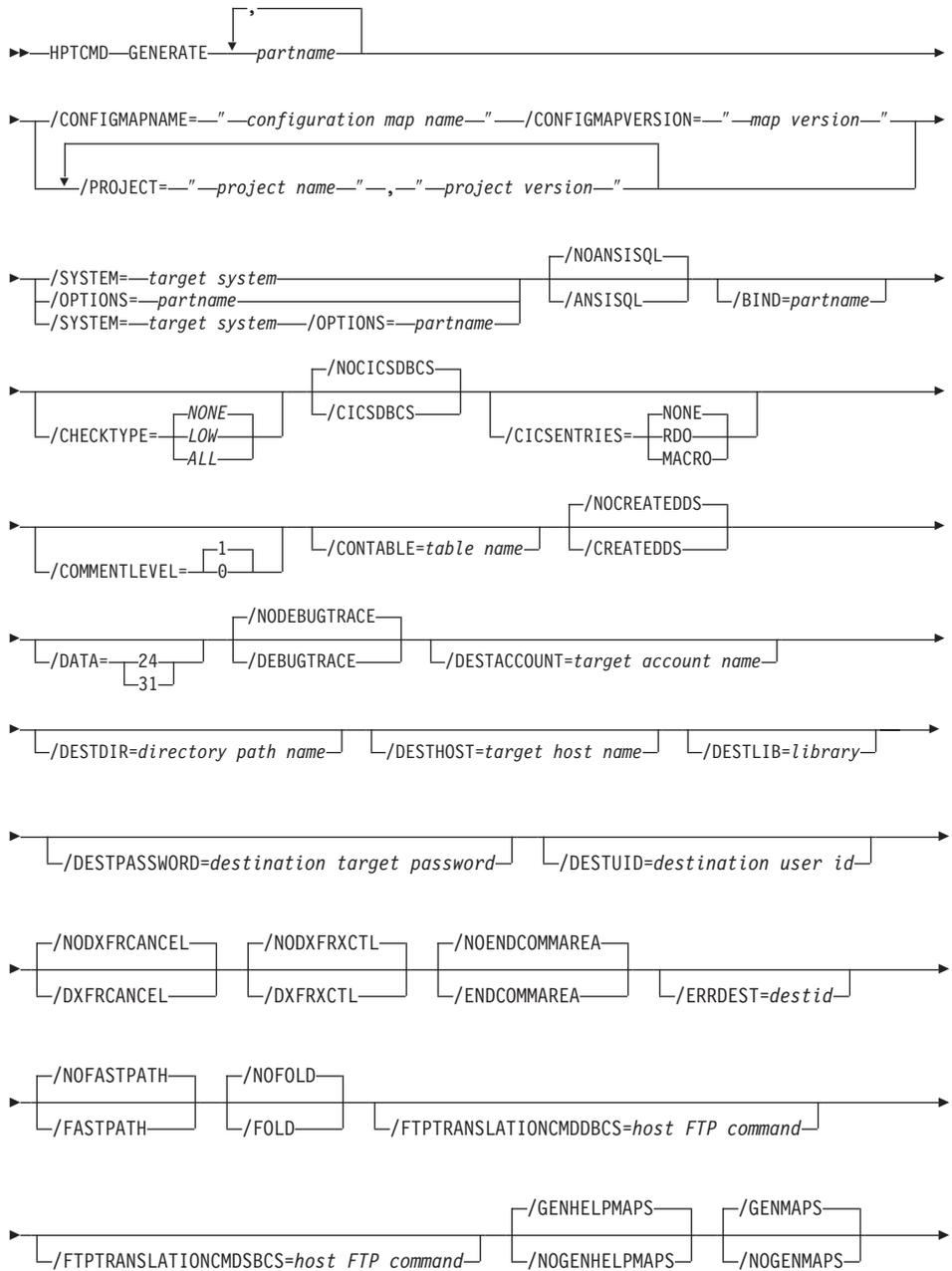
Note: If you are generating a program that uses DBCS, you must run the commands on a machine that is DBCS-enabled.

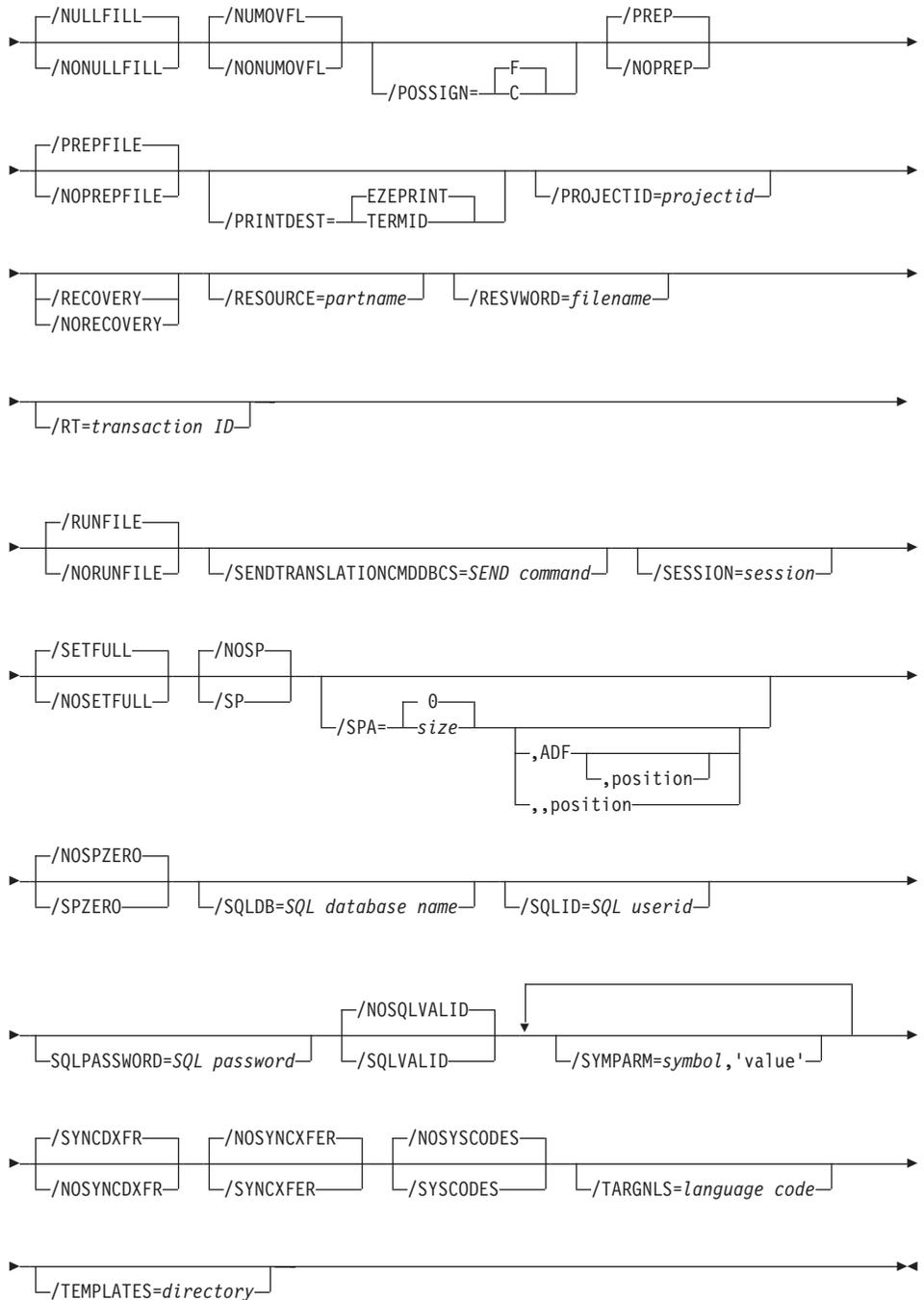
GENERATE subcommand syntax for COBOL generation

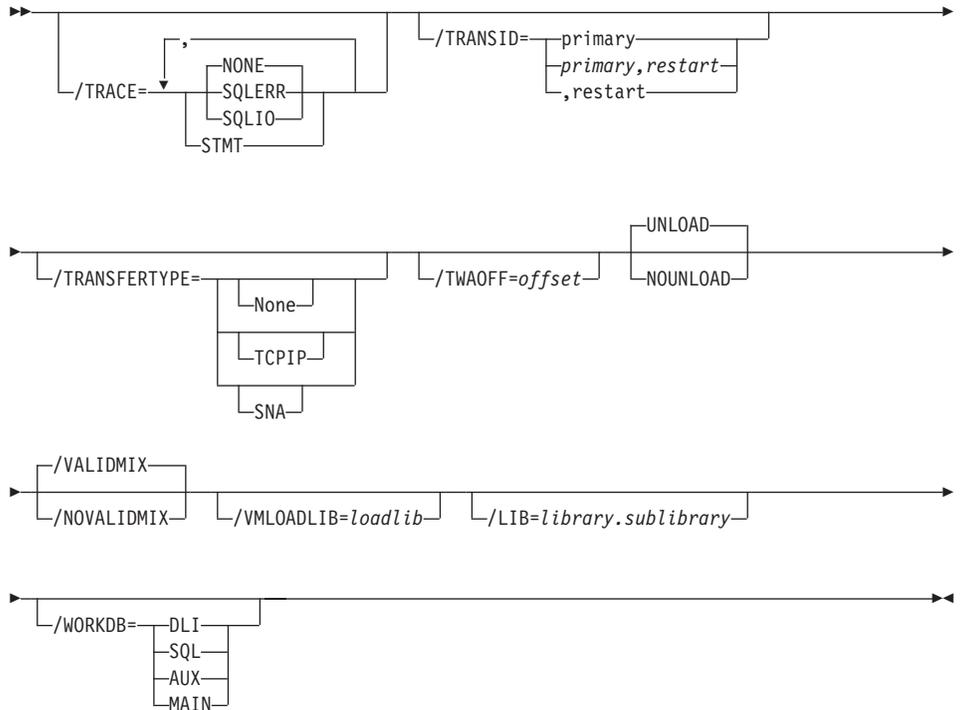
You use the GENERATE subcommand to generate a program, table, or map group. You can also specify options that affect how a part is generated.

The following syntax diagram shows the required command syntax. The subsequent section contains the syntax for each option that you can also specify on this command.

Note: The `/CONFIGMAPNAME`, `/CONFIGMAPVERSION`, `/PROJECT`, and `/SYSTEM` options require a value when you generate from the user interface.







See “Chapter 29. Generation command and option descriptions” on page 297 for the detailed information about the options illustrated in the syntax diagram.

GENERATE subcommand examples

This section contains examples of different uses of the GENERATE subcommand. Each example shows the GENERATE syntax used, along with associated information. Associated information can be the generation options part and resource associations. The examples appear on several lines because of the formatting of this document, but they must be entered as one command line.

Generating for CICS for MVS/ESA

Figure 52 on page 172 shows an example for generating a program for the CICS for MVS/ESA environment.

```
HPTCMD GENERATE PRGM1 /PROJECT="myproject","27.8"
/SYSTEM=MVSCICS
/PROJECTID=USER1
/JOBCARD=D:\JOBCARD.JCL
/TEMPLATES=C:\EFKGEN /SYMPARM=DSYS,'DSNA'
/LINKAGE=LINKMVSCICS
```

Figure 52. Generating for CICS for MVS/ESA

If you are generating with VisualAge Generator on Smalltalk, use /CONFIGMAPNAME and /CONFIGMAPVERSION instead of /PROJECT.

Generating for MVS batch

Figure 53 shows an example for generating a program for use in the MVS batch environment.

```
HPTCMD GENERATE PRGM1 /CONFIGMAPNAME="myconfig" /CONFIGMAPVERSION="27.8"
/SYSTEM=MVSBATCH
/PREP
/OPTIONS=OPTIONSMBAT
/RESOURCE=WORKRAP
/GENOUT=D:\WORK\GENOUTMBAT
```

Figure 53. Generating for MVS batch

Figure 54 shows the contents of the specified options part, OPTIONSMBAT.

```
/NOFOLD /LINES=55 /NOSQLVALID /NOLOCVALID /SESSION=A
/GENMAPS /GENHELPMAPS /GENTABLES /JOBCARD=EFK2MJOB.JCL
/PROJECTID=USER1 /LISTING
/SYMPARM=COB2LIB,'SYS1.COB2LIB'
/MSP=SEQ /SYMPARM=ELA,'ELA.V3R1M0'
```

Figure 54. OPTIONSMBAT

Generating for MVS/TSO

Figure 55 shows an example for generating a program for use in the MVS/TSO environment.

```
HPTCMD GENERATE A24X105 /CONFIGMAPNAME="myconfig" /CONFIGMAPVERSION="27.8"
/SYSTEM=TSO
/OPTIONS=A24X105
```

Figure 55. Generating for MVS/TSO

Figure 56 on page 173 shows the contents of the specified generation options part, A24X105.

```

/MSP=SEQ
/CONFIGMAPNAME="myconfig" /CONFIGMAPVERSION="27.8"
/PREP
/SESSION=B
/GENMAPS
/GENTABLES
/TEMPLATES=C:\EFKGEN
/JOBCARD=EFK2MJOB.JCL
/RESOURCE=A24X105
/PROJECTID=USER1
/GENOUT=D:\WORK\A24X105\TSO
/SYMPARM=ELA, 'ELA.V3R1M0'

```

Figure 56. A24X105

Figure 57 shows the contents of the specified resource association file, A24X105.

```

ASSOCIATE FILE=FXRA01 /SYSNAME='USER1.TSOMSL75' /FILETYPE=VSAMRS /SYSTEM=TSO
ASSOCIATE FILE=FRRA01 /SYSNAME='USER1.TSOMSL75' /FILETYPE=VSAMRS /SYSTEM=TSO
ASSOCIATE FILE=FSRA01 /SYSNAME='USER1.TSOMSL75' /FILETYPE=VSAMRS /SYSTEM=TSO

```

Figure 57. A24X105

Generating for CICS for OS/2

Figure 58 shows an example for generating a program for use in the CICS for OS/2 environment.

```

HPTCMD GENERATE PRGM1 /CONFIGMAPNAME="myconfig" /CONFIGMAPVERSION="27.8"
/SYSTEM=OS2CICS
/TEMPLATES=C:\EFKGEN
/LINKAGE=LINKCICSOS2

```

Figure 58. Generating for the CICS for OS/2 environment

Generating for VM CMS

Figure 59 shows an example for generating a program for use in the VM CMS environment.

```

HPTCMD GENERATE PRGM1 /CONFIGMAPNAME="myconfig" /CONFIGMAPVERSION="27.8"
/SYSTEM=VMCMS
/OPTIONS=OPTIONSVMCMS

```

Figure 59. Generating for VM CMS

Figure 60 on page 174 shows the contents of the specified generation options part, OPTIONSVMCMS.

```

/ DATA=24
/ NOSYNCFER
/ SYSTEM=VMCMS
/ SYMPARM=VMFMODE, 'C'
/ VMLoadLIB=APPL1

```

Figure 60. OPTIONSVMCMS

Generating for VM batch

Figure 61 shows an example for generating a program for use in the VM batch environment.

```

HPTCMD GENERATE PRGM1 /CONFIGMAPNAME="myconfig" /CONFIGMAPVERSION="27.8"
/ SYSTEM=VMBATCH
/ DATA /NOSYNCFER /SYMPARM=VMFMODE, 'D'
/ SYMPARM=SQLPKGNM, 'SQLAPPL' /SYMPARM=SQLSTMDE, 'MULTIUSER'
/ SYMPARM=SQLDBNAM, 'APPL1DB' /SYMPARM=SQLSRPW, 'SQLSRID/SQLPWD'

```

Figure 61. Generating for the VM batch environment

Generating for VSE batch

Figure 62 shows an example for generating a program for use in the VSE batch environment. In this example, the SQL program is running in single-user mode, and has print maps associated with the program.

```

HPTCMD GENERATE PRGM1 /CONFIGMAPNAME="myconfig" /CONFIGMAPVERSION="27.8"
/ SYSTEM=VSEBATCH
/ NOFOLD /LINES=55 /SQLVALID /NOLOCVALID /SESSION=A
/ GENMAPS /GENTABLES /JOB CARD=EFK0VJOB.JCL
/ LISTING /LIB=PRD2.USERLIB /MSP=SEQ
/ SYMPARM=SQLSTMDE, 'SINGLEUSER' /SYMPARM=VUSERLIB, 'PRD5.PROJLIB'
/ SYMPARM=SQLPKGNM, 'SQLAPPL' /SYMPARM=PROCLIB, 'PRD5.PROCLIB'
/ SYMPARM=PWRCLASS, '5'

```

Figure 62. Generating for the VSE batch environment

Generating for CICS for VSE/ESA

Figure 63 on page 175 shows an example for generating a program for use in the CICS for VSE/ESA environment.

```

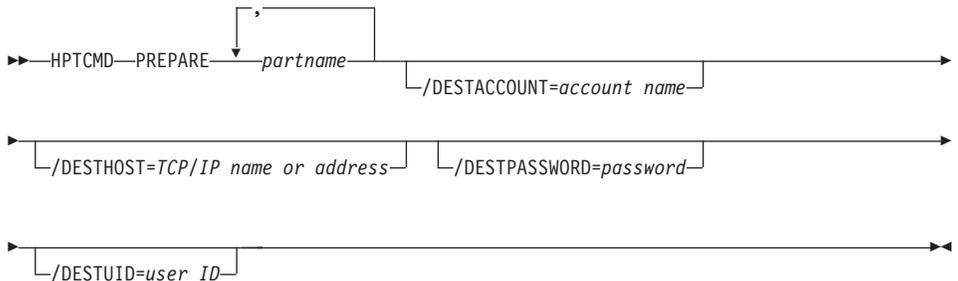
HPTCMD GENERATE PRGM1 /CONFIGMAPNAME="myconfig" /CONFIGMAPVERSION="27.8"
/SYSTEM=VSECICS
/PROJECTID=USER1
/LINKAGE=LINKVSECICS
/TEMPLATES=C:\EFKGEN
/GENOUT=D:\AG2\GENOUT
/JOBCARD=D:\AG2\JOBCARD.VSE
/RESOURCE=VSECICS
/SESSION=C
/PREP
/PREPROFILE
/GENMAPS
/GENTABLES
/NUMOVFL
/SYMPARM=PWRCLASS,'5'
/SYMPARM=NODUSRID,'HOST,USER1'
/LIB=PRD5.CUSTVAL
/CICSENTRIES=RDO
/SYMPARM=CRSLIB,'PRD5.CRS220B'
/SYMPARM=PROCLIB,'PRD2.EZELIB'
/SYMPARM=SQLUSRPW,'FIN/FIN'

```

Figure 63. Generating for the CICS for VSE/ESA environment

PREPARE subcommand syntax for COBOL generation

The PREPARE subcommand calls a program to interpret the preparation script (*partname.PRP*), which is created when the /PREPROFILE option is specified during generation.



See “Chapter 29. Generation command and option descriptions” on page 297 for the detailed information about the syntax diagram.

You can also start the preparation process by specifying the /PREP generation option. If the /PREPROFILE generation option is specified without the /PREP generation option, the PREPARE subcommand is used to prepare the generation output. The PREPARE subcommand is also used to restart

preparation if the preparation process is not successful in a manner that does not require the parts to be generated again.

The PREPARE subcommand causes different results, depending on the environment.

- For MVS and VSE programs, files are transferred from the workstation to the host using a script contained in the *partname.PRP* file. The *partname.PRP* file also contains a script to submit the preparation JCL generated for the program.
- For OS/400 programs, files are transferred from the workstation to the OS/400 environment using the script contained in the *partname.PRP* file. The *partname.PRP* file also contains a script to submit the preparation job stream generated for the program.
- For VM and CICS for OS/2 programs, files are transferred using the script contained in the *partname.PRP* file. The *partname.PRP* file also contains the script to invoke the preparation REXX generated for the program.

Example

This example illustrates how to prepare generated part PRGM1. In this case, the GENERATE subcommand specified a value of D:\MYDIR\OUTPUT for the /GENOUT option:

```
HPTCMD PREPARE D:\MYDIR\OUTPUT\PRGM1
```

Note: The values returned by the PREPARE subcommand only indicate the status of the preparation process itself. The values do not indicate whether any jobs submitted by the PREPARE subcommand are successful.

START subcommand syntax

The START subcommand starts the server process that runs HPTCMD subcommands.

▶—HPTCMD—START—▶

See “Chapter 29. Generation command and option descriptions” on page 297 for the detailed information about the syntax diagram.

If you do not issue the START subcommand and the server process is not already running, any HPTCMD subcommand you issue, except for the STOP subcommand, starts the server process automatically.

Note: If you do not issue the START subcommand and you direct your output to a location other than STDOUT, the server process inherits the STDOUT location. This causes the STDOUT file to be locked by the server until an HPTCMD STOP command is issued. To avoid this, always issue an HPTCMD START command, either at the command line or by placing the command at the beginning of your command file.

Example

The following is an example of how to use the START subcommand:

```
HPTCMD START
```

STOP subcommand syntax

The STOP subcommand stops the server process that runs HPTCMD subcommands.

▶▶—HPTCMD—STOP—▶▶

See “Chapter 29. Generation command and option descriptions” on page 297 for the detailed information about the syntax diagram.

The server process continues to run until it receives a STOP subcommand or until the Generation Monitor window is closed. When you specify the STOP subcommand, the server process completes all previously issued subcommands before stopping. If the server process is stopped as a result of the Generation Monitor window closing, the stop occurs immediately, ending any subcommand currently being processed.

Example

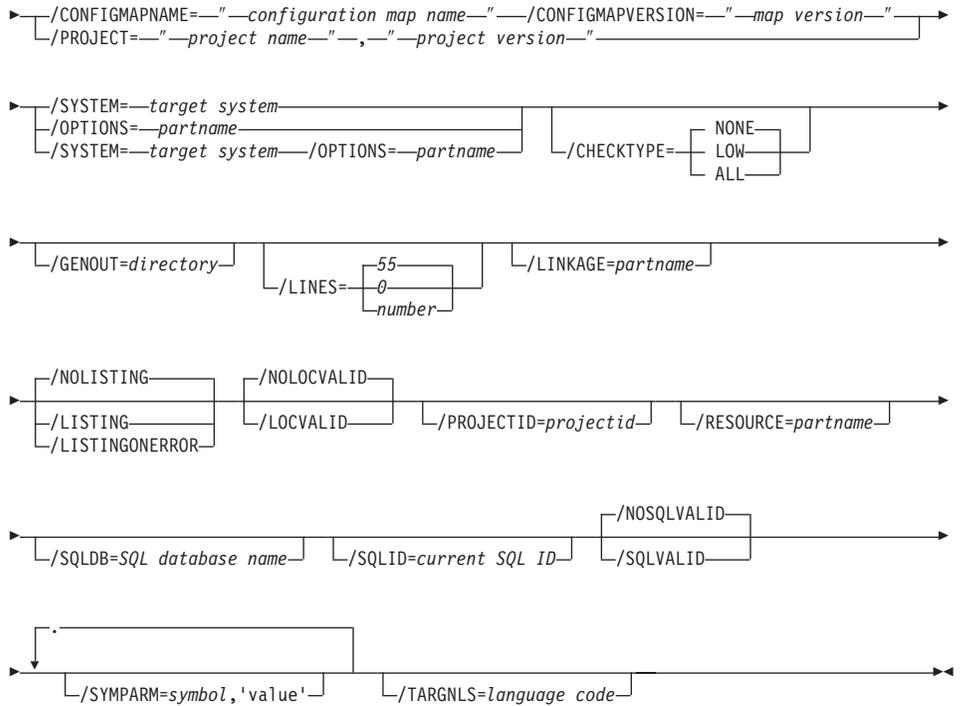
The following is an example of how to use the STOP subcommand:

```
HPTCMD STOP
```

VALIDATE subcommand syntax for COBOL generation

The VALIDATE subcommand enables you to validate your VisualAge Generator program without actually generating code.

▶▶—HPTCMD—VALIDATE—*partname*—▶▶



See “Chapter 29. Generation command and option descriptions” on page 297 for the detailed information about the syntax diagram.

Example

The following is an example of how to use the VALIDATE subcommand for COBOL validation. Enter the following as one continuous line at the OS/2 command prompt:

```
HPTCMD VALIDATE MYAPP
  /CONFIGMAPNAME="myconfig"
  /CONFIGMAPVERSION="27.8"
  /SYSTEM=OS2CICS
  /CHECKTYPE=ALL
  /SQLVALID
  /LOCVALID
```

Part 5. Generating Web transaction programs

Generating a Web transaction program produces both server source code in C++, Java, or COBOL, as well as Java gateway source code. Server code for Web transaction programs is created using the C++, Java, or COBOL generator.

Refer to Part 2. Generating C++ programs, Part 3. Generating Java programs, and Part 4. Generating COBOL programs. Java gateway code and related files are created when you generate a Web transaction program, a user interface record, or a table.

This part contains the following chapters:

- “Chapter 16. Inputs to Web transaction program generation” on page 181
- “Chapter 17. Outputs of Web transaction program generation” on page 183
- “Chapter 18. Preparation for Web transaction program generation” on page 185
- “Chapter 19. Command interface for Web transaction program generation” on page 187

For further details on Web transaction programs, see the Web Transaction Development Guide.

Chapter 16. Inputs to Web transaction program generation

Inputs to generating Web transaction programs include the inputs needed for C++, Java, or COBOL generation. For a complete list, refer to “Chapter 2. Inputs to C++ program generation” on page 11, “Chapter 6. Inputs to Java server program generation” on page 31, or “Chapter 10. Inputs to COBOL generation” on page 51.

When generating Web transaction programs, user interface records, or tables for the gateway, review the following generation options, as described in “Optional parameters for subcommands” on page 300:

- /GENAUTHORTIMEVALUES
- /GENOUT
- /GENRESOURCEBUNDLE
- /GENUIRECORDS
- /JVADESTHOST, /JVADESTUI, /JVADESTDIR, and /JVADESTPASSWORD: Specify those four if you want the Java code and related files to be transferred to a remote Java target machine. All four must be valid for the transfer to occur.
- /JAVASYSTEM
- /JSPRELDIR
- /MSGTABLEPREFIX
- /PACKAGENAME
- /RESOURCEBUNDLELOCALE
- /TARGNLS

Chapter 17. Outputs of Web transaction program generation

This chapter describes the Java gateway outputs of Web transaction program generation.

Outputs of generating Web transaction program parts

In addition to Java gateway outputs, outputs of generating Web transaction programs include the outputs produced by C++, Java, or COBOL generation. For a complete list, refer to “Chapter 3. Outputs of C++ program generation” on page 15, “Chapter 7. Outputs of Java program generation” on page 35, or “Chapter 13. Outputs of COBOL generation” on page 137.

Java gateway code and related files are stored in the /GENOUT directory on the generation machine and then transferred to the Java target machine if necessary.

Java output files are based on the name of the member being generated. However, if the member name contains characters other than alphanumeric characters and underscores, those characters are replaced with *xyyyy* where *yyyy* is the hexadecimal representation of the character. For example, files generated from a member named "UI-RCD" would contain "UIx002DRCD" in their names.

Java generation outputs

Table 48 identifies the Java outputs that result from generating Web transaction program parts.

Table 48. Java generation outputs

Element	File name and extension	Environment
Java source for each user interface record generated	VGUirxxxxxx.java ¹	All
Java source for each user interface record bean generated	xxxxxxBean.java	All
Java source for each resource bundle generated for a UI record when /GENRESOURCEBUNDLE is specified	<i>uirname</i> RBBundle.java ² or <i>uirname</i> RBBundle_locale.java ³	All
Java source for each resource bundle generated for a message table	<i>pppp</i> RBBundle.java ⁴ or <i>pppp</i> RBBundle_locale.java ³	All

Table 48. Java generation outputs (continued)

Element	File name and extension	Environment
Java source for each edit table generated	VGtblxxxxxx.java	All
Binary table contents for each edit table generated	VGDataxxxxxx.tab	All
Java source for each edit function local-storage record generated	VGWkgxxxxxx.java	All
JSP for each user interface record generated	xxxxxx.jsp	All
Preparation file	xxxxxxJ.cmd	All
File transfer protocol (FTP) control file	xxxxxxJ.ftp	All
Batch command file that invokes the Java compiler on either a local or remote target machine	xxxxxxJZ.bat (for Windows NT) or xxxxxxJZ.cmd (for OS/2)	Windows NT, OS/2
Batch command file for a member generated for a local target machine; copies files to the package directory prior to compilation	xxxxxxLZ.bat (for Windows NT) or xxxxxxLZ.cmd (for OS/2)	Windows NT, OS/2
Script file that invokes the Java compiler on the remote target machine	xxxxxxJ.scr	Any UNIX platform; or OS/400

Notes:

- ¹ xxxxxxx indicates a 7-character member name.
- ² uirname indicates a UI record name.
- ³ locale indicates a Java locale of the form ll, ll_ll, or ll_ll_ll, where l is a letter.
- ⁴ pppp indicates a 4-character message table prefix.

Chapter 18. Preparation for Web transaction program generation

This chapter describes how to prepare the Java gateway parts of a Web transaction program. For information on how to prepare the C++, Java, or COBOL parts, see “Chapter 4. Preparation process for C++ generation” on page 17, “Chapter 8. Preparation process for Java generation” on page 39, or “Chapter 14. Preparation process for COBOL generation” on page 153.

When generation is complete, preparation takes place automatically unless you specify the /NOPREP option. The preparation process comprises the following steps:

1. Transferring parts to the Java target environment, if needed
2. Unicode conversion, if needed
3. Compiling the Java code

Preparation requirements

If you want the Java compiler classes to be copied to somewhere other than to the location specified by /JAVADESTDIR or /GENOUT, you must set the HPTCLASSDIR and HPTJSPDIR environment variables. HPTCLASSDIR is the root of the Java package directory to which you want to deploy your tab files and Java class files on your Java target system. It should be included in your classpath. Thus, if my.pkg is the package name then HPTCLASSDIR should be set to the directory that contains the \my\pkg directory. HPTJSPDIR is the directory to which you want to deploy your JSP files.

A .cmd file is generated for each Web transaction program, user interface record, or Java table being generated. That file uses FCEJBLD.EXE to control preparation processing.

See “Chapter 31. Analyzing return codes and errors” on page 367 for information about preparation errors.

Preparing Web transaction Java parts

FCEJBLD.EXE does the following processing:

1. If the generation machine is different from the Java gateway machine, the xxxxxxj.ftp file is used to transfer all the required files to the Java gateway machine. For a Web transaction program or user interface record, this transfer includes the Java source for the following items:
 - UI record beans

- UI record objects
- Resource bundles
- Edit function local-storage record definitions
- Edit table definitions
- Edit table contents

This transfer also includes JSP files and the following, as appropriate:

- For OS/2, the files *xxxxxxxJZ.cmd* and *xxxxxxxLZ.cmd*
 - For OS/400 and the UNIX environments, the file *xxxxxxxj.scr*
 - For Windows NT, the files *xxxxxxxJZ.bat* and *xxxxxxxLZ.bat*
2. If the Java target system is a local Windows NT (or OS/2) machine, FCEJBLD.EXE runs the file *xxxxxxxLZ.bat* (or *xxxxxxxLZ.cmd*) to copy the Java code to the package directory, which is a subdirectory of /GENOUT. FCEJBLD.EXE then runs the *xxxxxxxJZ.bat* (or *xxxxxxxJZ.cmd*) to do the following:
- Perform Unicode conversion, if necessary
 - Compile the Java code
 - Copy the class and tab files to the directory HPTCLASSDIR
 - Copy the JSP files to the directory specified by HPTJSPDIR

If the Java target system is a remote Windows NT (or OS/2) machine, the file *xxxxxxxJZ.bat* (or *xxxxxxxJZ.cmd*) must be run manually.

If the Java target system is a UNIX environment or OS/400, FCEJBLD.EXE runs *xxxxxxxj.scr* to perform the same tasks as *xxxxxxxJZ.bat*.

Chapter 19. Command interface for Web transaction program generation

You can issue the VisualAge Generator Developer subcommands from a system prompt or from within a command file. The command HPTCMD implements the command interface. Subcommands are specified with the HPTCMD command and are followed by any required keywords and options. Comments can be imbedded in the commands. Comments begin with the characters `/*` and end with the characters `*/`.

Command processing can be started explicitly by issuing the START subcommand or implicitly by issuing any other VisualAge Generator Developer subcommand. The command continues running until it is ended, either by issuing the STOP subcommand or by closing the Generation Monitor window.

Starting the command opens a Generation Monitor window. The Generation Monitor window displays the command currently being processed and provides information showing what stage of generation the process has reached. You can cancel the currently processing command from the Generation Monitor window. Closing the Generation Monitor window ends any command currently being processed.

If you are generating programs using a generation server, refer to the *VisualAge Generator System Development Guide* for information about starting and stopping the Generation Monitor.

Note: If you are generating a program that uses DBCS, you must run the commands on a machine that is DBCS-enabled.

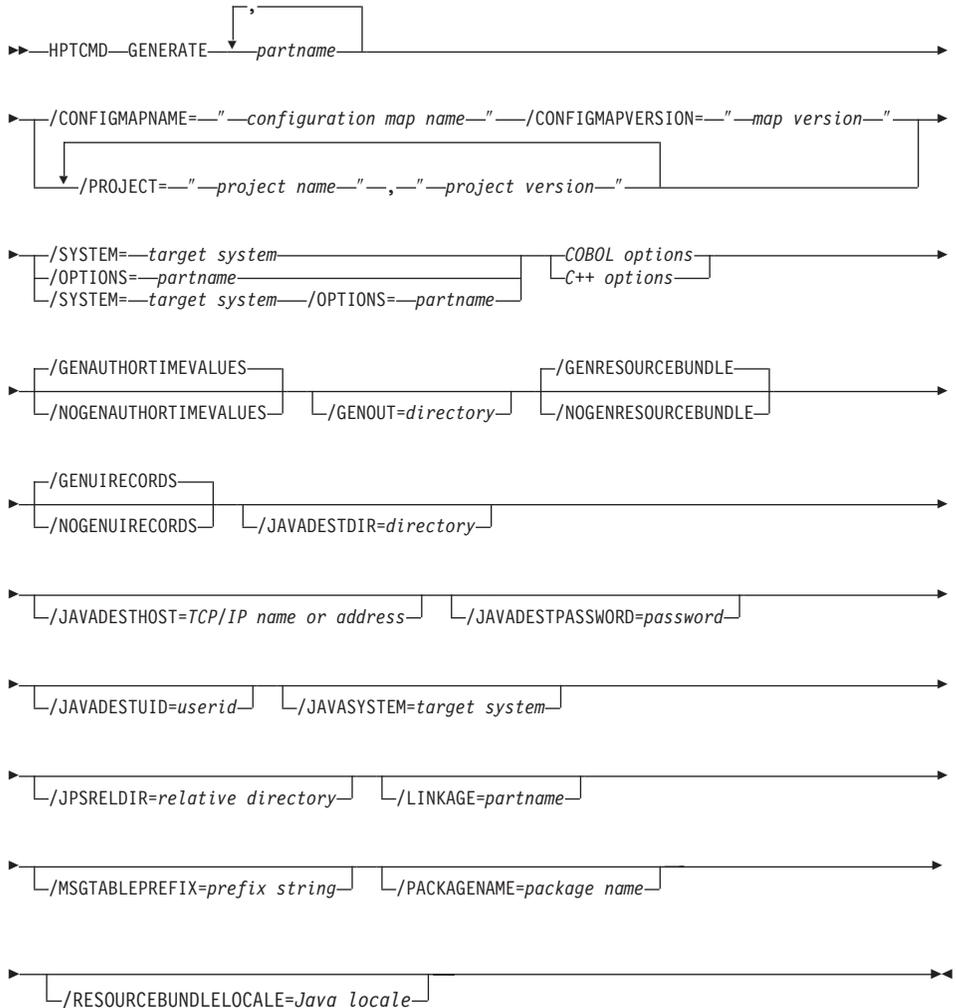
GENERATE subcommand syntax for Web transaction program generation

You use the GENERATE subcommand to generate a Web transaction program, user interface record, or table. You can also specify options that affect how a part is generated.

The following syntax diagram shows the options for the GENERATE subcommand for Web transaction program generation. If you are also generating COBOL code, see "GENERATE subcommand syntax for COBOL generation" on page 167. If you are also generating C++ code, see

“GENERATE subcommand syntax for C++ generation” on page 21. If you are also generating Java server code, see “GENERATE subcommand syntax for Java generation” on page 43.

Note: The /CONFIGMAPNAME, /CONFIGMAPVERSION, and /SYSTEM options require a value when you generate from the command interface by using VisualAge Generator on Smalltalk. The /PROJECT and /SYSTEM options require a value when you generate from the command interface using VisualAge Generator on Java.



See “Chapter 29. Generation command and option descriptions” on page 297 for the detailed information about the syntax diagram.

GENERATE subcommand example for Web transaction

Figure 64 illustrates the GENERATE subcommand for Java program generation; at an OS/2 command prompt it must be entered in one continuous line.

```
HPTCMD GENERATE MYPGM
/CONFIGMAPNAME="myconfig" /CONFIGMAPVERSION="27.8"
/SYSTEM=WINNT
/COMMENTLEVEL=0;
/DESTDIR='\\u\pancho'
/DESTHOST=quixote.rtp.ibm.com
/DETPASSWORD=mypassword
/DESTUID=pancho
/OPTIONS=MYOPT
/GENOUT=D:\pancho\gen
/TRACE=PROCESSES
/JAVADESTDIR='/home/jasper/javagen/genout'
/JAVADESTHOST=quixote.rtp.ibm.com
/JAVADETPASSWORD=javapw
/JAVADESTUID=don
/JAVASYSTEM=AIX
/PACKAGENAME=my.pkg
```

Figure 64. GENERATE subcommand for Java

The last six lines set specifications for generation of Java parts for the GatewayServlet. The others pertain to C++ server generation.

The AIX, HP-UX, and Solaris environments are case sensitive.

For information about setting generation options, see “Chapter 24. Generation options parts” on page 209.

For information about generation return codes, see “Chapter 31. Analyzing return codes and errors” on page 367.

Syntax of other HPTCMD subcommands

For information about the command interface for other HPTCMD subcommands, see “Chapter 15. Command interface for COBOL generation” on page 167 or “Chapter 5. Command interface for C++ generation” on page 21.

Part 6. Generating JavaBeans wrappers and session beans

This part describes how to generate JavaBeans wrappers and session beans. A session bean is a type of Enterprise Java Bean (EJB).

The ability to generate JavaBeans wrappers for a server program is available only in VisualAge Generator Developer on Java. Because VisualAge Generator Developer on Java is not available for the OS/2 platform, JavaBeans wrappers can be generated only in a Windows NT development environment. Generated JavaBeans wrappers can be run on any supported client platform.

An Enterprise Java Bean session bean can be generated to invoke a server program.

This part contains the following chapters:

- “Chapter 20. Inputs to Java wrapper generation” on page 193
- “Chapter 21. Outputs of Java wrapper generation” on page 195
- “Chapter 22. Command interface for Java wrapper generation” on page 201
- “Chapter 23. Generating session beans” on page 205

Chapter 20. Inputs to Java wrapper generation

The Java program generation process uses inputs from several different sources:

- VisualAge Generator part definitions that define the program, and its associates.

Refer to the VisualAge Generator Developer online help system for more information about defining parts.

- Generation control files and parts, which are predefined. The generation control files and parts for Java are described in the following sections:

- “Linkage table parts” on page 53

Refer to the *VisualAge Generator Client/Server Communications Guide* document for information about linkage tables.

- “Chapter 24. Generation options parts” on page 209

These control parts and files contain installation and project-level conventions that specify how the Java program’s wrappers are generated. Generation control parts and files are usually specified at an installation or project level, but they can be specified at any level. You can modify the contents of the generation control parts and files.

Inputs for JavaBeans wrapper generation

JavaBeans wrappers can be generated for VisualAge Generator server (called batch) programs. The inputs to wrapper generation are the program member and the members defined in the called parameter list. All parameter types are supported except for maps.

Generation options

To generate JavaBeans wrapper classes for a VisualAge Generator server (called batch) program, specify `/SYSTEM=JAVAWRAPPER` as the target run-time system when generating the server program.

Following are other options that you can specify when generating Java wrappers:

- `/EJBGROUP=`
- `/LINKAGE=`
- `/OPTIONS=`
- `/PACKAGENAME=`
- `/PROJECT=`

The following is an example of a batch generation command for generating JavaBeans wrappers for program STAFFMN:

```
hptcmd generate staffmn /PROJECT="Staff","R45.0" /SYSTEM=JAVAWRAPPER  
/PACKAGENAME=StaffPkg /LINKAGE=staff.lkg
```

This generation statement assumes that all parts required by program STAFFMN are in the Staff project. If there are referenced parts in more than one project, there must be a /PROJECT option specified for each project required. (If generating wrappers from the interactive generation user interface, all referenced parts must be in projects loaded in the image.) The wrappers would be generated into the StaffPkg package. The package specified by the /PACKAGENAME generation option must be contained in a project named by a /PROJECT option, and the edition of the package in the named project must be an open edition.

Linkage table options

The linkage table entry for the called batch program defines how the program should be called from the JavaBeans wrapper. Linkage table options are supported as documented in the *VisualAge Generator Client/Server Communications Guide* except as noted in the following sections.

Generation or run-time binding for linkage options

If REMOTEBIND=GENERATION is specified for the server program when the Java wrapper is generated, the linkage table options are generated into the server program wrapper.

If REMOTEBIND=RUNTIME is specified, only the program name and linkage table name are generated into the server program wrapper. The linkage table is read again at run time on the Web server (for applets), on the client system (for programs), or on the Web application server for servlets and JavaServer Pages (JSPs).

Chapter 21. Outputs of Java wrapper generation

VisualAge Generator generates beans for wrapping calls to the server program. The following types of classes are generated:

- “Beans for servers” on page 196
- “Beans for record parameters” on page 198
- “Beans for record array rows” on page 199

Each generated class is stored in the VisualAge for Java repository in the package with the name specified by the /PACKAGENAME generation option. The package name must exist. For batch generation, the package name must be contained by a project named by a /PROJECT option. For interactive generation, the package must be contained in a project loaded into the current image. The edition of the package in the project specified must be an open edition.

The developer can use the Javadoc tool to build a classname.html file once the the class has been compiled. The HTML file describes the public interfaces for the class.

The following sections give a short description of what is generated for each class. The descriptions include the HTML descriptions of the classes built for server program STAFFMN. Figure 65 on page 196 lists the VisualAge Generator External Source Format (ESF) definition of STAFFMN from which the classes were generated.

```

:appl      name = STAFFMN   type = CALLBATCH.
:mainprc  name = STAFFMN-MAIN.
:emainprc.
:callparm name = BUTTON-PRESSED type = ITEM.
:callparm name = STAFF-MAINT   type = RECORD.
:eappl.
:record   name = STAFF-MAINT   org = WORKSTOR scope = LOCAL.
:recditem name = ROWS-FETCHED  type = NUM   bytes=4.
:recditem name = SQL-CODE      type = NUM   bytes=8.
:recditem name = FETCH-LIMIT   type = NUM   bytes=8.
:recditem name = STAFF-DATA    type = CHA   bytes=28 occurs=10.
:recditem name = ID            type = BIN   bytes=2 level=20.
:recditem name = NAME          type = CHA   bytes=9 level=20.
:recditem name = DEPT          type = BIN   bytes=2 level=20.
:recditem name = JOB           type = CHA   bytes=5 level=20.
:recditem name = YEARS         type = BIN   bytes=2 level=20.
:recditem name = SALARY        type = PACK  bytes=4 decimals=2 level=20.
:recditem name = COMM          type = PACK  bytes=4 decimals=2 level=20.
:erecord.
:item     name = BUTTON-PRESSED type = CHA   bytes=1.
:eitem.

```

Figure 65. ESF for example server program STAFFMN

Beans for servers

A bean generated for a server program includes the following:

- The private instance variables for each parameter. The Java data type for each item parameter is generated as shown in Table 49 on page 197.
- The get and set methods for each parameter. For data item parameters, the set methods signal parameter changes using the `PropertyChange` event as defined for beans. If other beans are "listening" for changes in parameters, you must use the set methods to change the parameters rather than directly assigning new values to them.
- An **execute** method for calling the server program using the class instance variables of the server program wrapper as parameters.
- A **call** method for calling the server with an argument defined corresponding to each parameter in the program called parameter list.
- The **AddPropertyChangeListener** and **removePropertyChangeListener** methods enable signaling of other beans when parameter values are changed by a set method or on return from a server call. These methods are inherited from the `CSOServerProgram` class.

The developer can use the server program as a bean by using the set methods to set parameter values prior to calling the server, the execute method to call the server program, and the get methods to retrieve the returned parameter

values after calling the server program. The listener methods enable another bean to be notified whenever the data values are changed.

The developer can also treat the server call as a function call, passing the parameters as arguments on the call method. If the call method is used, then get methods must be used to retrieve the values returned for data item parameters, since Java primitive parameters are always passed by value.

When /PACKAGENAME=StaffPkg is specified at generation, Class **StaffPkg.Staffmn** is the server class generated for sample program STAFFMN. For more information, see the *VisualAge Generator Client/Server Communications Guide*.

Table 49 shows the derivation of Java data types from VisualAge Generator item definitions.

Table 49. Derivation of Java data types from VisualAge Generator item definitions

VisualAge Generator Data Type	Length in chars or digits	Length in bytes	Decimals	Java Data Type	Maximum precision in Java
CHA	1-32767	1-32767	NA	String	NA
MIX	1-32767	1-32767	NA	String	NA
DBCS	1-16383	1-32767	NA	String	NA
UNICODE	1-16383	1-32767	NA	String	NA
HEX	2-75534	1-32767	NA	Byte[]	NA
BIN	1-4	2	0	Short	4
BIN	5-9	4	0	Int	9
BIN	10-18	8	0	Long	18
BIN	1-4	2	>0	Float	4
BIN	5-9	4	>0	Double	15
BIN	10-18	8	>0	Double	15
NUM, NUMC	1-4	1-4	0	Short	4
NUM, NUMC	5-9	5-9	0	Int	9
NUM, NUMC	10-18	10-18	0	Long	18
NUM, NUMC	1-6	1-6	>0	Float	6
NUM, NUMC	7-18	7-18	>0	Double	15

Table 49. Derivation of Java data types from VisualAge Generator item definitions (continued)

VisualAge Generator Data Type	Length in chars or digits	Length in bytes	Decimals	Java Data Type	Maximum precision in Java
PACK, PACF	1-3	1-2	0	Short	4
PACK, PACF	4-9	3-5	0	Int	9
PACK, PACF	10-18	6-10	0	Long	18
PACK, PACF	1-5	1-3	>0	Float	6
PACK, PACF	7-18	4-10	>0	Double	15

Beans for record parameters

A bean generated for a record includes the following:

- Public instance variables, such as the following:
 - Java primitives for each low-level item that is not within a substructured array.
 - Java primitive arrays for item arrays
 - An object array for each substructured array item

The Java data type associated with each record item type is shown in Table 49 on page 197.

- Get and Set methods for each instance variable, enabling the record class to be used as a JavaBean. The Set methods signal instance variable changes using the PropertyChange event as defined for JavaBeans. If other JavaBeans are "listening" for changes in parameters, you must use the Set methods to change the instance variables rather than directly assigning new values to them.
- **AddPropertyChangeListener** and **removePropertyChangeListener** methods that enable signaling of other beans when parameter values are changed by a set method or on return from a server call. These methods are inherited from the Record class.
- Nonpublic methods for marshalling record data on a server call occurring structure in the record on a server call.

If a parameter is an SQL row record, then get and set methods are generated to allow you to query and set the null indicator for each data item of the record. For example, for a data item named PHONE, the method `getPhoneNullIndicator()` queries whether the value for PHONE is null.

When `/PACKAGENAME=StaffPkg` is specified at generation, class **StaffPkg.StaffMaint** is the record class generated for record parameter

STAFF_MAINT in the sample program STAFFMN (see Figure 65 on page 196). For an example of the Java documentation generated for a record class, see the file `staffmnP.StaffMaint.html` in the `com.ibm.vgj.cso` package documentation. For more information, see the *VisualAge Generator Client/Server Communications Guide*.

Beans for record array rows

A bean generated for a multiply occurring substructure in a parameter record includes the following:

- Public instance variables defined as Java primitives for each low-level item in the substructure. The Java data type associated with each record item type is shown in Table 49 on page 197.
- Get and Set methods for each instance variable, allowing the class to be used as a `JavaBean`. The Set methods signal instance variable changes using the `PropertyChange` event as defined for `JavaBeans`. If other `JavaBeans` are "listening" for changes in parameters, you must use the Set methods to change the instance variables rather than directly assigning new values to them.
- If a parameter is an SQL row record, then get and set methods are generated to allow you to query and set the null indicator for each data item of the record. For example, for a data item named `PHONE`, the method `getPhoneNullIndicator()` queries whether the value for `PHONE` is null.
- The `AddPropertyChangeListener` and `removePropertyChangeListener` methods enable signaling of other beans when parameter values are changed by a set method or on return from a server program call. These methods are inherited from the `CSORecord` class.
- Non-public methods for marshalling record data on a server call.

When `/PACKAGENAME=StaffPkg` is specified at generation, class `StaffPkg.StaffMaint_StaffData` is the record array row class generated for multiply occurring substructure `STAFF_DATA` in record `STAFF_MAINT`. For more information, see the *VisualAge Generator Client/Server Communications Guide*.

Chapter 22. Command interface for Java wrapper generation

You can issue the VisualAge Generator Developer subcommands from a system prompt or from within a command file. The command HPTCMD implements the command interface. Subcommands are specified with the HPTCMD command and are followed by any required keywords and options. Comments can be imbedded in the commands. Comments begin with the characters `/*` and end with the characters `*/`.

Command processing can be started explicitly by issuing the `START` subcommand or implicitly by issuing any other VisualAge Generator Developer subcommand. The command continues running until it is ended, either by issuing the `STOP` subcommand or by closing the Generation Monitor window.

Starting the command opens a Generation Monitor window. The Generation Monitor window displays the command currently being processed and provides information showing what stage of generation the process has reached. You can cancel the currently processing command from the Generation Monitor window. Closing the Generation Monitor window ends any command currently being processed.

If you are generating programs using a generation server, refer to the *VisualAge Generator System Development Guide* for information about starting and stopping the Generation Monitor.

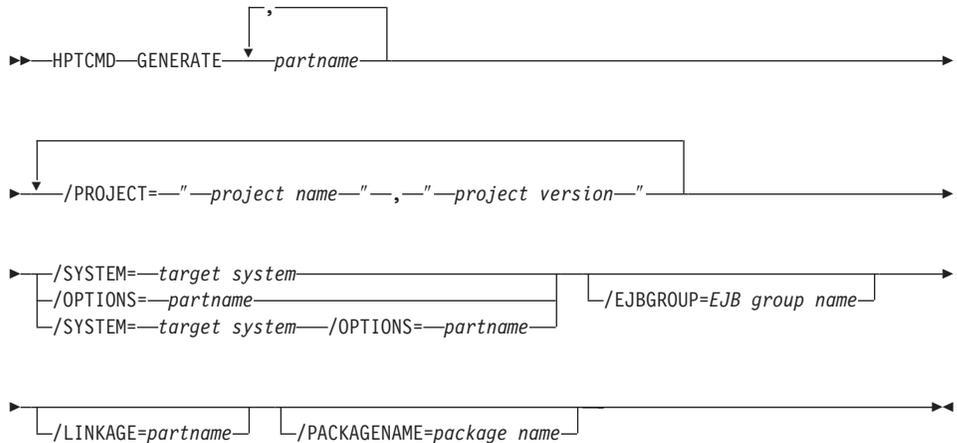
Note: If you are generating a program that uses DBCS, you must run the commands on a machine that is DBCS-enabled.

GENERATE subcommand syntax for Java wrapper generation

You use the `GENERATE` subcommand to generate a program, table, or map group. You can also specify options that affect how a part is generated.

The following syntax diagram shows the options for the `GENERATE` subcommand.

Note: The `/SYSTEM` option requires a value when you generate from the user interface. (You must specify a target environment from a drop-down menu before you can generate code.)



See “Chapter 29. Generation command and option descriptions” on page 297 for the detailed information about the syntax diagram.

Example

Figure 66 illustrates the GENERATE subcommand for Java program generation; at an OS/2 command prompt it must be entered in one continuous line.

```

HPTCMD GENERATE MYPGM
/PROJECT="myconfig", "27.8"
/SYSTEM=JAVAWRAPPER
/OPTIONS=MYOPT

```

Figure 66. GENERATE subcommand for Java

See “Chapter 24. Generation options parts” on page 209 for information about setting generation options.

See “Chapter 31. Analyzing return codes and errors” on page 367 for information about generation return codes.

START subcommand syntax

The START subcommand starts the server process that runs HPTCMD subcommands.

```

HPTCMD START

```

See “Chapter 29. Generation command and option descriptions” on page 297 for the detailed information about the syntax diagram.

If you do not issue the START subcommand and the server process is not already running, any HPTCMD subcommand you issue, except for the STOP subcommand, starts the server process automatically.

Note: If you do not issue the START subcommand and you direct your output to a location other than STDOUT, the server process inherits the STDOUT location. This causes the STDOUT file to be locked by the server until an HPTCMD STOP command is issued. To avoid this, always issue an HPTCMD START command, either at the command line or by placing the command at the beginning of your command file.

Example

The following is an example of how to use the START subcommand:

```
HPTCMD START
```

STOP subcommand syntax

The STOP subcommand stops the server process that runs HPTCMD subcommands.

▶▶—HPTCMD—STOP—————▶▶

See “Chapter 29. Generation command and option descriptions” on page 297 for the detailed information about the syntax diagram.

The server process continues to run until it receives a STOP subcommand or until the Generation Monitor window is closed. When you specify the STOP subcommand, the server process completes all previously issued subcommands before stopping. If the server process is stopped as a result of the Generation Monitor window closing, the stop occurs immediately, ending any subcommand currently being processed.

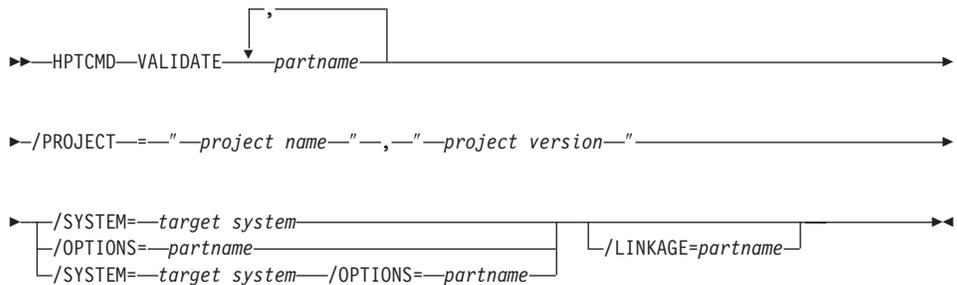
Example

The following is an example of how to use the STOP subcommand:

```
HPTCMD STOP
```

VALIDATE subcommand syntax for Java wrapper generation

The VALIDATE subcommand enables you to validate your VisualAge Generator program without actually generating code.



See “Chapter 29. Generation command and option descriptions” on page 297 for the detailed information about the syntax diagram.

Example

Figure 67 illustrates how to use the VALIDATE subcommand for Java wrapper validation.

```
HPTCMD VALIDATE MYGEN /PROJECT="myconfig", "27.8" /SYSTEM=JAVAWRAPPER
```

Figure 67. Java wrapper validation

Chapter 23. Generating session beans

A session bean is a type of Enterprise Java Bean (EJB). To generate a session bean to invoke a server program, specify the following generation options:

1. /SYSTEM=JAVAWRAPPER
2. /EJBGROUP=*ejbgroup* where *ejbgroup* is the name of an EJB group created in VisualAge for Java. The EJB group must exist, and the loaded edition of it must be an open edition.
3. /LINKAGE=*linkageTableName* where *linkageTableName* is the name of a linkage table part containing an entry for the server program that is being generated. The linkage table entry for the program must meet the following criteria:
 - a. It must specify the LINKTYPE=SESSIONEJB attribute. If this attribute is not specified, only Java wrappers for the program and its record parameters will be generated.
 - b. It may specify the providerURL attribute to identify the Java Naming and Directory Interface (JNDI) server to be used to locate the session bean when deployed. This attribute is required only if the JNDI name server resides on a different machine than the client that invokes the session bean.
 - c. It may specify REMOTEBIND=GENERATION if the generated server program wrapper is used to invoke the session bean, the server program will always be invoked through a session bean, and it is never necessary to determine at run time other linkage attributes such as where the server program resides and how to invoke it. There is a performance benefit in not having to read a linkage table at run time to figure out how to implement a call.

For information on using Java wrappers and session beans and on deploying session beans in an enterprise Java server, refer to the *VisualAge Generator Client/Server Communications Guide*.

Part 7. Reference information

Chapter 24. Generation options parts

Generation options enable you to customize part generation. You can specify generation options using the following methods:

- By selecting values before generation by using the Options notebook in the VisualAge Generator Developer product
- As options on the batch command when you are using the GENERATE subcommand
- By setting values for these options in generation options parts

Generation options are stored in generation options parts, which enable developers working on the same project to share generation options. You can specify a generation options part using the /OPTIONS option on the GENERATE subcommand.

Creating generation options parts

You can create and edit a generation options part using the Generation Options Editor in VisualAge Generator Developer. You can enter options and values in this editor in uppercase or lowercase. The entries in a generation options part can extend across multiple lines, and more than one generation option can be specified on a line.

Comments are permitted anywhere in the part. Begin comments with the characters /* and end them with the characters */.

Option values can be specified only one time in a generation options part, except for the /SYMPARM option. Any subsequent specification of an option within the same generation options part is ignored, and a message is not issued. The /SYMPARM option can be specified multiple times in a generation options part.

The example below shows a sample generation options part.

```
/* PROJECT: CSAT - CUSTOMER SATISFACTION SURVEY */  
/* VisualAge Generator GENERATOR OPTIONS DEFAULTS PART */  
/MATH=COBOL /PROJECTID=CSAT /NONUMOVFL  
/LOCVALID /LISTINGONERROR  
/COMMENTLEVEL=INFO  
/SETFULL
```

The generation options part is specified using the /OPTIONS generation option.

Making the default generation options part available during generation

You can use one of the following methods to make the default generation options part available to the program during generation:

1. Create a package/application and release it into a configuration map or project that is to contain parts that are to be used for each generation. Do one of the following:
 - Specify the configuration map as a required map of each configuration map used for generating a program, table, or map group.
 - Specify the project used for generating a program, table, or map group.
2. Copy and save the workspace/image used to load the VisualAge Generator Developer feature after the initial load. Develop the default generation options part in a package/application, and save this workspace/image. Use the workspace/image in which the default generation options part was saved to do each generation.
3. Create the default generation options part in a package/application and, before each generation, load the package/application containing the default generation options part into the workspace/image used for generation before invoking the generator.

Establishing default generation options

The default generation options part is a special type of generation options part. The default generation options part has two main purposes:

- Setting default generation values other than those predefined by VisualAge Generator Developer
- Restricting the use of specific generation options

Default generation options parts with NOOVERRIDE

The NOOVERRIDE parameter can follow the specification of any option in a default generation options part. If NOOVERRIDE is specified for an option, the value of that option cannot be overridden. If an attempt is made to define the option value again, a message is issued.

Using the NOOVERRIDE parameter in the default generation options part enables the system administrator or project leader to restrict the ability to change the definition of default generation options. If NOOVERRIDE is not specified for an option, the value of that option can be overridden. You can specify values for the option in other generation options parts or through the VisualAge Generator Developer user interface. You can also specify values for options using the GENERATE subcommand.

You can specify a generation options part as the default options part by setting the *defaultGenerationOptions* value in the *hpt.ini* file.

For example, the system administrator might want the program developers to always use the `/MATH=CSPA` option. The system administrator can create a generation options default file to contain the default generation option specification for the `/MATH` option. By specifying `NOOVERRIDE` for the `/MATH` option, the system administrator prevents any later definition of the `/MATH` option from replacing the default definition. Suppose the system administrator creates a generation options defaults file and names it `DEFAULTOPT`. The example below shows the definition in the default generation options part with `NOOVERRIDE`.

```
/MATH=CSPA NOOVERRIDE
```

The system administrator instructs the developers to specify `DEFAULTOPT` for the *defaultGenerationOptions* value in the `hpt.ini` file. Because the `/MATH` option is defined with the `NOOVERRIDE` parameter, any later values specified for `/MATH` are not accepted, and a message is issued.

Default generation option part without `NOOVERRIDE`

Generation options defined in the default generation options part without the `NOOVERRIDE` parameter can be overridden.

For example, your system administrator might want developers to use a specific linkage table. The system administrator can specify the linkage table name in the default generation options part, but still permit program developers to override this specification. Suppose the system administrator creates a generation options part and names it `DEFAULTOPT`. The recommended linkage table for development is `DEFAULTLKG`. The example below shows a sample linkage table specification in the default generation options part.

```
/LINKAGE=DEFAULTLKG
```

The system administrator instructs the developers to specify `DEFAULTOPT` for the *defaultGenerationOptions* value in the `hpt.ini` file. However, because the `/LINKAGE` option does not have `NOOVERRIDE` specified, developers can specify a different linkage table, or they can specify `/LINKAGE=""` if they do not want to use any linkage table.

By specifying a different value for the `/LINKAGE` option in a generation options part or by using the `GENERATE` subcommand itself, developers can override the linkage table specified in the default generation options part. The example below shows a sample generation options default file. In this example, the system administrator defined values for the `/MATH`, `/PROJECTID`, `/NUMOVFL`, `/OPTIONS`, `/INITADDWS`, and `/INITRECD` generation options. All other generation options default to values set by the VisualAge Generator Developer. The `/MATH`, `/OPTIONS`, `/INITADDWS`, and

/INITRECD options are defined with the NOOVERRIDE parameter. If a developer specifies values for these options, the option specification is ignored, and a message is issued.

```
/* PROJECT: CSAT - CUSTOMER SATISFACTION SURVEY */  
/* VisualAge Generator GENERATOR OPTIONS DEFAULTS PART */  
/MATH=COBOL NOOVERRIDE  
/PROJECTID=CSAT /NONUMOVFL  
/OPTIONS=SYSOPTS NOOVERRIDE  
/INITADDWS NOOVERRIDE /INITRECD NOOVERRIDE
```

Using multiple levels of generation options parts

Generation options parts can contain default options for a company, a project, an environment, a single program, or a user. Multiple levels of options files can be set up so that options can be overridden by options specified in higher-level files.

For example, suppose the system administrator has set up a default generation options part, IMSOPT, to contain the company standard generation options for IMS programs. The project leader for the new project, P1, wants to use all the standard options for IMS, except the project leader wants the developers to store the generated objects in project-related directories and partitioned data sets. The example below shows a sample options part named IMSOFTP1 created by the project leader.

```
/GENOUT=W:\P1 /PROJECTID=P1 /OPTIONS=IMSOPT
```

The project leader instructs the developers to specify the generation options part IMSOFTP1 when generating IMS programs for project P1. The /GENOUT and /PROJECTID options override the corresponding options in IMSOPT. All other options values are used from IMSOPT. The generated objects are stored in the P1 directory. Prepared objects on the MVS host are stored in partitioned data sets with the P1 project qualifier.

Note: In this case, the system administrator decided not to use a generation options defaults part. If the system administrator had instructed the developers to specify IMSOPT for the *defaultGenerationOptions* value in the *hpt.ini* file, IMSOPT would be the generation options defaults part for the project. Then the project leader would not have to set the /OPTIONS generation option in the IMSOFTP1 options file to point to IMSOPT.

Determining generation option resolution order

When there are multiple specifications for the same option, the value used is determined in the following sequence:

1. Options defined in the *defaultGenerationOptions* part specified in the *hpt.ini* file if NOOVERRIDE is specified for the option

2. Options explicitly selected in the Generation Options notebook
3. Options from one of the following:
 - a. Generation Options part explicitly specified in the Generation Options notebook
 - b. Generation Options part explicitly specified in the VisualAge Preferences notebook
4. Options from generation options parts specified using the /OPTIONS chain
5. Options from the defaultGeneraitonOptions part specified in the *hpt.ini* file if NOOVERRIDE is not specified for the option

Within a single generation options part, the first-found specification for an option is used.

See “Establishing default generation options” on page 210 for more information on using the NOOVERRIDE parameter.

Using the sample generation options parts

During product installation, sample environment-related generation options parts are placed in the directory C:\Program Files\Vast. You can use these generation options parts and generation options defaults files as a basis for creating generation options parts that are customized to meet the specific needs of your organization.

The sample generation options parts are templates for you to modify. The values specified for the options in the sample generation options parts are usually the default values. If you do not want to change the values from the default values listed, you do not need to use options parts.

Sample generation options default part

Table 50 shows the sample generation options default files.

Table 50. Sample generation options default files

File name	Description
EZEOPVAL.OPT	Sample default options part for the validation phase
EFKOPDFT.OPT	Sample default options for the production phase

Sample generation options parts

Table 51 on page 214 shows the sample generation options parts. These names might have been modified during product installation to meet the needs of your organization.

To make these parts available to your program, read the file in using the **Generation Options Part** editor.

Table 51. Sample generation options parts

Environment	File name
AIX	FCEDFLT.OPT
CICS for AIX	FCEDFLT.OPT
HP-UX	FCEDFLT.OPT
IMS BMP, including /MFSDEV options that require customization	EFKOPBMP.OPT
IMS/VS, including /MFSDEV options that require customization	EFKOPIMS.OPT
MVS batch	EFKOPBAT.OPT
CICS for MVS/ESA	EFKOPMCS.OPT
MVS/TSO	EFKOPTSO.OPT
CICS for OS/2	EFKOPOCS.OPT
OS/2 (C++)	FCEDFLT.OPT
OS/400	EFKOPOS4.OPT
VM CMS	EFKOPVM.OPT
VM batch	EFKOPVMB.OPT
VSE batch	EFKOPVBA.OPT
CICS for VSE/ESA	EFKOPVCS.OPT
Windows NT	FCEDFLT.OPT
CICS for Windows NT	FCEDFLT.OPT

Using symbolic parameters in generation option specifications

Many of the generation option values are file names, directory names, or identifiers. You can specify symbolic parameters as part of these option values. The symbolic parameter delimiter is the percent sign (%). However, if you are specifying a symbolic parameter in a command file, you must use two percent signs (%%) as the delimiter. The example below shows the /GENOUT generation option if you specify the symbolic parameter EZEENV as part of the value in a generation options part.

```
/GENOUT=D:\MYOUT\%EZEENV%
```

The example below shows the /GENOUT generation option if you specify the symbolic parameter EZEENV as part of the value in a command file.

```
/GENOUT=D:\MYOUT\%%EZEENV%%
```

Generation options that are not valid

Options that do not apply to the target system are ignored, and a message is not generated. Messages are only generated when the value specified for an option is not valid, regardless of whether the option itself applies to the target system.

Overriding a value to use the default value

The following example shows how to specify an option to set it back to the default value.

```
/LINKEDIT= ''
```

This setting specifies that any previous value set for the `/LINKEDIT` option is not to be used and that the default value is to be used instead.

Guidelines for setting generation options

Except for the AIX, HP-UX, SCO, and Solaris environments, you must use the following guidelines when specifying a value for an option that requires a name, such as a file or directory name:

- The first character must be one of the following:
 - The letters a–z or A–Z
 - The numbers 0–9
 - The characters @ # \$ *
- Characters other than the first one can be any character that is valid for a file name except the following:
 - , () or a space character

Note: If you are specifying a file or directory name that contains these characters, you must enclose the value in single quotation marks (') or double quotation marks (").

For the AIX, HP-UX, SCO, and Solaris environments, you must use the following guidelines when specifying a value for generation options:

- The AIX, HP-UX, SCO, and Solaris environments are case sensitive. Verify the case for the specified values for the following options:
 - `/DBPASSWORD`
 - `/DBUSER`
 - `/DESTDIR`
 - `/DESTHOST`
 - `/DESTUID`
 - `/DETPASSWORD`
- The AIX, HP-UX, SCO, and Solaris environments use a slash in path names. Use the following guidelines when specifying a value for `/DESTDIR`:

- Verify the case.
- Enclose the value in single quotes.
- If you are specifying a fully qualified path, begin with a forward slash.

The following example shows how to specify /DESTDIR in the AIX, HP-UX, SCO, and Solaris environments.

```
/DESTDIR='/u/mydir/genout'
```

Chapter 25. Linkage tables

A linkage table is an optional control file used during generation, test, and by the client run-time environments. The entries in the table control the following functions:

- CALLLINK** Specifies linkage conventions to be used for calling a program.
- CRTXLINK** Specifies whether a CICS CREATX service call starts a local or remote CICS transaction.
- DXFRLINK** Specifies linkage conventions to be used for implementing a DXFR transfer between host programs.
- FILELINK** Specifies whether a CICS file is to be accessed as a local or remote file.

The file name for the linkage table file used during test is specified on the test facility general preferences window. The file used during generation is specified using the /LINKAGE generation option.

Creating a linkage table

Use a standard editor or the Repository/ENVY library to create a linkage table. In this file or part, you can enter data in uppercase or lowercase. The syntax of the file is validated when the file is used by test or generation.

The entries in the linkage table file are coded using a tag language. Valid tags are CALLLINK, CRTXLINK, DXFRLINK, and FILELINK. Each tag has attributes that enable you to modify the tag. You can enter attributes for a tag in any order.

Tags and attributes can appear on separate lines or you can have multiple entries per line. The same attribute cannot be specified more than once for a tag.

Comments begin with the characters /* and end with the characters */.

If the program or file name specified in a linkage table entry ends with an asterisk, the entry applies to all programs or files whose name begins with the characters preceding the asterisk. For example, consider a program name specified as:

```
:calllink applname=myapl* ...
```

This table entry is used for any call to a program and for any called program that has MYAPL as the first 5 characters of the program name.

If multiple entries are valid for a name, the first table entry that matches the name is used. For example, consider a program name specified as:

```
:calllink  
applname=* ...
```

This entry applies to all called programs. You can use an entry like this following all other :calllink entries to override the default linkage options for all :calllink entries.

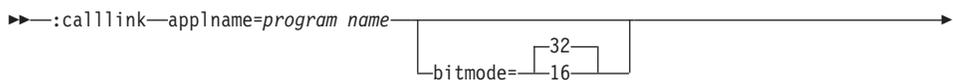
Specifying an asterisk is not allowed for program names on the :dxfrlink statement.

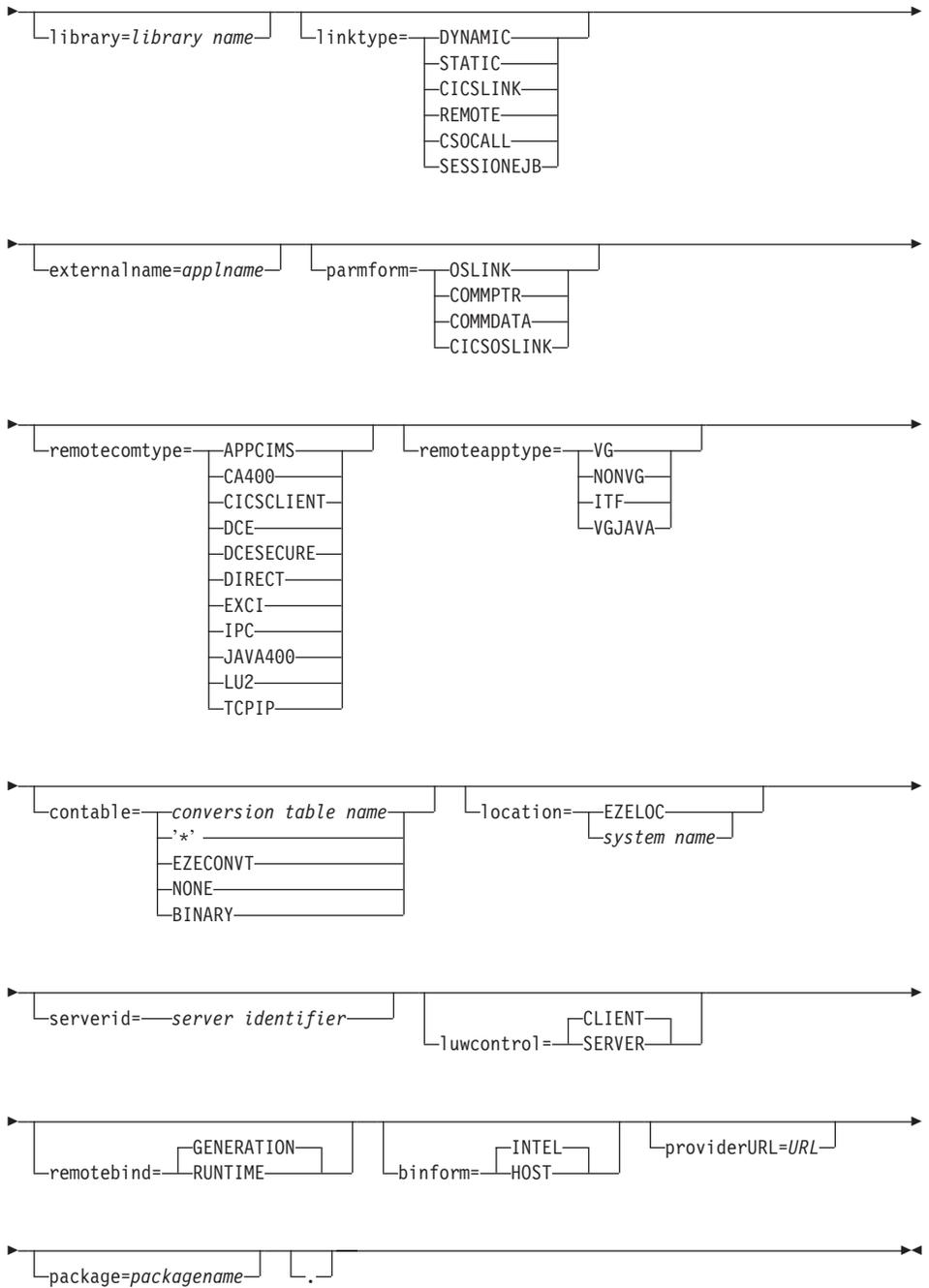
Specifying CALL linkage (CALLLINK)

The calllink tag specifies the type of linkage to be used for a call from one program to another. The tag affects how test facility and client run times call an external program and how the generator generates both the called and calling programs.

Note: In a GUI program, a call can be defined by coding a CALL statement or by drawing an event-to-action connection from an event to the execute action of a callable function object that represents the called (target) program. The linkage table description for CALL statements from GUI programs applies to both types of calls. If the linkage information is specified in the properties of the event-to-action connection, then the linkage information from the properties is used instead of the linkage table.

The following diagram shows the attributes you can specify for the calllink tag:





Definitions for CALLINK

aplname

Specifies the name of the called program as specified on the CALL statement in the calling program.

You can use an asterisk (*) as a global substitution character in the program name parameter; however, it is only valid as the last or only character.

bitmode

Specifies whether a called DLL program runs in 16- or 32-bit mode.

This option is effective only for DLL programs (LINKTYPE=DYNAMIC PARMFORM=OSLINK) called from GUI programs.

The default is 32-bit mode.

Calls from C++ programs are always done in 32-bit mode. Test facility dynamically determines whether a DLL uses 16- or 32-bit mode prior to making the call.

library

Specifies the name of the library that contains the called program.

This option is effective only for programs (linktype=DYNAMIC parmform=OSLINK) called from GUI or C++ programs or the test facility, or for remote calls to OS/400 servers.

The meaning of the name depends on the called program environment as shown in Table 52:

Table 52. Linktype for Called Program Environment

Linktype	Called Program Environment	Library Type
DYNAMIC	OS/2, Windows	DLL
DYNAMIC	AIX, HP-UX, Solaris	Shared library
REMOTE	OS/400	OS/400 program library
CSOCALL	OS/400 (with Java400 protocol)	OS/400 program library
CSOCALL, remotecomtype=DIRECT	OS/2, Windows	DLL
CSOCALL, remotecomtype=DIRECT	AIX, HP-UX, Solaris	Shared library

The default value is the program name.

Note: In prior releases, the keyword was DLLNAME instead of the system independent term LIBRARY. For compatibility, DLLNAME is treated as a synonym for LIBRARY.

linktype

Specifies the type of linkage being generated.

DYNAMIC

For generated COBOL program calls, specifies that a dynamic COBOL call is to be performed.

For calls from GUI, C++ or programs or the test facility, specifies that a standard C call to a DLL is to be performed. A run-time error occurs if the DLL specified in the applname attribute cannot be loaded. A search for an executable (CMD or EXE file) is *not* done. If the applname attribute specified is not a valid DLL name because it contains an asterisk (*), specify the library attribute to limit the search to DLLs only.

For calls from Java server programs, a call to a local Java server program is to be performed.

This value is the default value for non-CICS environments.

STATIC

For generated COBOL programs call, specifies that a static COBOL call is to be performed

For calls from C++ programs, Java programs, or the test facility, treated like linktype=DYNAMIC.

This value is required for calls to PL/I programs in non-CICS host environments.

CICSLINK

For calls from CICS programs, specifies that an EXEC CICS command is to be performed.

This value is the default value for programs generated for CICS environments and is valid only for CICS environments. This value is required for calls to PL/I programs in the CICS environment.

CSOCALL and REMOTE

Specifies that the program is a remote or local program and the call is routed through the VisualAge Generator middleware.

The remote program (LINKTYPE=REMOTE) and the csocall program (LINKTYPE=CSOCALL) are equivalent for all

platforms except Java GUIs. CSOCALL may be used instead of REMOTE wherever REMOTE is valid. Java GUIs can only specify CSOCALL.

CSOCALL and REMOTE are valid only for calls to server programs from CICS, GUI, C++ or Java programs, or from the test facility. CSOCALL and REMOTE are typically used when the called program runs on a different system, thus utilizing the middleware to provide the necessary communications and any data conversion between the different systems. If the client and server reside on the same system, the middleware provides a variety of benefits such as the ability to call a VisualAge Generator server program residing in the Repository/ENVY library, the capability of each call to be an independent server-controlled unit of work, or increased control at run time via the linkage table.

If the caller is a CICS program, a CICS distributed program link statement is performed with actual data being passed in the CICS COMMAREA control block.

If the caller is a GUI, C++, or Java program, or the test facility, the protocol used is determined by the REMOTECOMTYPE attribute.

SESSIONEJB

Specifies that the program is to be invoked from a session bean generated for the program. The generated session bean must be deployed on an enterprise Java server, and the name server used to locate the session bean must either reside on the same machine as the enterprise Java server, or the PROVIDERURL attribute must be used to identify where the name server resides. In order for a client to make a call to a server program through a session bean, the client must be generated using a linkage table whose entry for the server program contains the LINKTYPE=SESSIONEJB attribute. You cannot wait until run time to specify this attribute in a run-time linkage table. LINKTYPE=SESSIONEJB is only applicable for Java clients calling server programs.

When LINKTYPE=SESSIONEJB is specified, middleware Java classes locate the session bean and invoke its call method. The session bean then calls the server program just as if LINKTYPE=CSOCALL had been specified. If REMOTEBIND=RUNTIME is specified, the session bean attempts to locate the linkage table on the machine where it is

deployed to obtain information about where the server program resides and on what protocol to use to pass data to the program.

externalname

The name of the program that is called. The name must be equal to the APPLNAME in all cases except for DYNAMIC or STATIC calls from C++ programs and for remote calls to OS/400 servers. The EXTERNALNAME value can be more than 8 characters long in these cases.

The default value is the APPLNAME.

parmform

Specifies the format in which parameters are passed on the call.

For remote programs (linktype=REMOTE or linktype=CSOCALL), parameter format is determined by the run-time protocol. The format is OSLINK for OS/400 servers accessed using CA/400 and for IMS servers. Otherwise, the format is COMMDATA for remote programs. Either COMMDATA or COMMPTR can be specified for calls to non-VisualAge Generator CICS programs.

OSLINK

Specifies that the program parameters are passed using standard parameter passing conventions.

If a DLL or load module is being called, parameters are passed by reference. If a CMD or EXE is being called, the parameters are moved to the command line buffer separated by blanks.

When OSLINK is specified for CICS environments, the called program must not contain any CICS commands and the called program cannot be a generated COBOL program.

COMMPTR

Specifies that the program parameters are passed using a pointer list in the COMMAREA consisting of one 4-byte pointer for each parameter.

In MVS CICS and VSE CICS systems, the high-order bit of the last pointer to set to 1.

For CICS OS/2, CONTABLE=NONE is required when COMMPTR is used.

This value is valid only in CICS environments. This value is the default value for CICS environments for the DYNAMIC, STATIC, and CICS LINK link types.

COMMDATA

Specifies that the parameters are passed in a single buffer. In CICS programs the buffer is in the COMMAREA

Each parameter value is moved to the buffer adjoining the previous value without regard for boundary alignment. If variable length records are passed, space is reserved for the maximum length defined for the record. If a variable length record with a record length item is passed, the item must be defined within the variable length record structure.

The called program returns the parameter values in the COMMAREA in the same order in which it received them. The calling program moves the returned parameter values in the COMMAREA back to the original parameters.

The EZEDLPSB and EZEDLPCB parameters receive special handling. When EZEDLPSB is passed, the 12-byte PSB structure (PSB name plus user interface block (UIB) pointer) is moved to the buffer. If EZEDLPCB is passed, a four-byte pointer to the PCB is moved to the buffer.

The COMMDATA value is valid for calls to CICS and remote programs.

CICSOSLINK

Specifies that the parameters are passed by reference using standard COBOL parameter passing conventions. The CICS EIB and COMMAREA are always passed as the first two parameters followed by the program parameters.

This value is valid only in CICS environments and only with a link type of STATIC or DYNAMIC. For MVS environments, DYNAMIC linkage can only be used with CICS Version 3 or later systems.

remotecomtype

Specifies the communication protocol for use by GUI clients, C++ programs, Java programs, or the test facility for calls to remote programs.

APPCIMS	LU 6.2 connection to IMS message processing region
CA400	Client Access/400
CICSCLIENT	CICS Client ECI
DCE	Unauthenticated Distributed Computing Environment (DCE) Remote Procedure Call (RPC). Use only if in the server program is not designated as a secure program.

DCESECURE	Authenticated DCE RPC call. The server checks whether the client's active DCE login identifier is authorized to run the server program. There is extra overhead associated with performing authenticated RPC calls so DCESECURE should not be used unless it is necessary. Valid only for OS/2, Windows NT, and AIX platforms.
DIRECT	VisualAge Generator middleware using a direct local call for client and server programs residing on the same system. The advantages of using this protocol are documented along with the client configuration information where the protocol is supported. For example, see the <i>VisualAge Generator Client/Server Communications Guide</i> .
EXCI	Extended CICS Interface is used to start MVS CICS transactions from generated Java wrappers. Use this value only when calling server programs from Java applets or Java applications. For details on starting MVS CICS transactions from Java wrappers, see the <i>VisualAge Generator Client/Server Communications Guide</i> .
IPC	VisualAge Generator VisualAge Generator middleware using inter-process communications for client and server programs residing on the same system. The advantages of using this protocol are documented along with the client configuration information where the protocol is supported. For example, see the <i>VisualAge Generator Client/Server Communications Guide</i> .
Java400	Protocol to connect to remote AS/400 servers by using AS/400 Toolbox for Java. Use this value only when calling a server program from a Java applet or Java GUI applications. For more information, see the <i>VisualAge Generator Client/Server Communications Guide</i> .
LU2	VisualAge Generator middleware using the LU2 protocol
TCPIP	VisualAge Generator middleware using the TCP/IP protocol

remoteapptype

Specifies whether a remote called program is a generated VisualAge Generator program, a program developed using another program development tool, or a VisualAge Generator program residing in the Repository/ENVY library.

VG The remote procedure is a generated VisualAge

	Generator program. An additional parameter is passed to allow the server to notify the client program if the server program ends abnormally.
VGJAVA	The remote procedure is a generated VisualAge Generator Java program.
NONVG	The remote procedure is a program developed using a tool other than VisualAge Generator. Only the parameters specified on the call statement are passed.
ITF	The remote procedure is a VisualAge Generator program residing in the Repository/ENVY library that will run under the control of the test facility.

contable

Specifies the name of the conversion table used to perform automatic data conversion on a remote CALL statement from a client to a remote server program. Anytime that you are coming from a Java client or your client and server platforms are different, you will need to specify a conversion table. For additional information, see the *VisualAge Generator Client/Server Communications Guide*.

The attribute is supported if the linkage type is specified as REMOTE or CSOCALL for CICS programs, or REMOTE, CSOCALL or CICSLINK for clients running under the test facility.

conversion table name

Conversion is performed on the client using the conversion table specified.

EZECONVT

Conversion is performed on the client using the conversion table name in the EZECONVT special function word at run time. If EZECONVT contains blanks, no conversion is performed.

NONE

No conversion is performed. This is the default value if VisualAge Generator communication middleware is not used.

CONTABLE=NONE is required for calls to CICS OS/2 if PARMFORM=COMMPTR.

Some conversion table names have special meaning:

- * Conversion is performed on the client using the default conversion table. The default conversion table performs ASCII to EBCDIC character conversion, as well as binary numeric conversion. For additional information, see the *VisualAge Generator Client/Server Communications Guide*.

On OS/2, AIX, HP-UX, and Windows systems the default conversion table is the conversion table specified in the environment variable EZERCVT. The default conversion table used is based on the generation system, the target run-time environment, and the locale (country/language) specified at generation time. For additional information on the conversion tables shipped with VisualAge Generator, see the *VisualAge Generator Client/Server Communications Guide*. If EZERNLS is not specified, the default code is ENU.

On MVS or VSE systems, the default conversion table is ELAxxxxx where xxxxx is the code specified when the calling program was generated (TARGNLS generation option).

BINARY

Only binary fields are converted. The byte order in the binary field is reversed.

This table is used with OS/2 and Windows clients communicating with AIX, Solaris, and HP-UX servers, and vice versa, when both the client and the server are running under the same code page.

The default value is NONE.

location

Specifies how the location (system identifier) of a remote program is determined at run time.

This attribute value is used only if the linkage type is specified as REMOTE or CSOCALL.

EZELOC

Specifies that the system identifier for the remote program is obtained from the EZELOC special function word when a call is done to the program.

system name

The system identifier for the system on which the server program resides. The meaning of the system identifier varies with the protocol. The following table describes the meaning of the identifier by protocol and the default value if location is not specified.

Protocol	Meaning of location	Default value
APPCIMS	CPIC side information identifier. The side information specifies: <ul style="list-style-type: none"> • Partner LU Alias • Transaction Program Name • Mode Name 	No default
CA/400	AS/400 system identifier	The managing OS/400 system
CICS DPL	CICS system identifier	System identifier defined for applname in the CICS tables.
CICSCLIENT	CICS system identifier	First system identifier specified in the CICS client initialization file.
DCE, DCESECURE	Location where the server advertises in the DCE CDS database. The location is specified in the configuration file used when the VisualAge Generator DCE server program is started.	No default
Java400	AS/400 system identifier	The managing OS/400 system
TCP/IP	TCP/IP hostname	No default

serverid

Protocol dependent channel or transaction identifier associated with the server or VisualAge Generator communication middleware gateway

Values are as follows:

server identifier

The server identifier name to be used for this call. The meaning of the name varies with the communication protocol as shown in the following table:

Protocol	Meaning of Server Identifier
CICSCLIENT	Name of CICS transaction for the server. If client unit of work is specified, all programs called in the same unit of work must have the same server identifier. The default is the CICS server system mirror transaction. For DB2 server applications on CICS for MVS/ESA, an RCT entry is needed. The RCT is used for the serverid.
DCE, DCESECURE	Serverid name advertised by the server in the DCE CDS database. The serverid is specified in the configuration file used when the VisualAge Generator DCE server program is started.

Protocol	Meaning of Server Identifier
TCP/IP	Service name as defined in the TCP/IP services file. If the call is from a Java program (GUI or server), the <i>serverid</i> is the port number of the remote program's listener.

The *serverid* is ignored for protocols not listed in the table.

luwcontrol

Specifies whether client or server controls unit of work:

- CLIENT** Unit of work is under client control. Server updates are not committed or rolled back until the client requests commit or rollback. Server programs cannot call EZECOMIT or EZEROLLB. This is the default value, unless client controlled unit of work is not supported in the server environment.
- SERVER** Server unit of work is independent of the client's unit of work. Commit (or rollback on abnormal termination) are automatically issued when the server returns. Server programs can call EZECOMIT or EZEROLLB.

the *VisualAge Generator Client/Server Communications Guide* shows the environments that support client- and server-controlled units of work.

remotebind

Specifies when linkage options used for a call to a remote program are determined. Values are as follows:

GENERATION

Linkage used for the call statement is determined by the linkage table specified at generation. This is the default value.

RUNTIME

The linkage table is read at run time. Specified values in the run-time linkage table override the values specified at generation. Generation values are used if omitted from the run-time table.

Common client access looks for the linkage table on the current DPATH (OS/2, AIX) or PATH (Windows) search path. The linkage table name is the same as the linkage table file name specified at generation. If this table cannot be found, common client access looks for the file named in the environment variable CSOLINKTBL.

The generated program always passes EZECONVT and EZELOC contents to common client access in case the run-time values for CONTABLE and LOCATION require them. The program always notifies common client access of commits and rollbacks in case client controlled unit of work is requested at run time.

The remotebind option is supported for generated C++, Java, and GUI programs.

binform

This option is supported only for the test facility. The preprocessor step does not pass this information to the generation process.

The value specifies in what format the user passes binary fields.

HOST Specifies that the user passes binary fields in host format.

INTEL

Specifies that the user passes binary fields in Intel format. The default is Intel.

providerURL

This property specifies the host name and port of the name server used by a Java client to locate a session bean that calls a server program.

URL The property value must have the following format:
iiop://hostname:port, where *hostname* is the IP address or host name of the machine on which the name server runs and *port* is the port number on which the name server listens.

This attribute is applicable only if linktype=SESSIONEJB is also specified, and if the client is a Java client. Since most URLs contain periods, and may contain a port number preceded by a colon, the URL specified should be enclosed in double quotes.

For example, the property value "iiop://bankserver.mybank.com:9019" directs an EJB client to look for a name server on the host named bankserver.mybank.com listening on port 9019. The property value "iiop://bankserver.mybank.com" directs an EJB client to look for a name server on the host named bankserver.mybank.com at port number 900. The property value "iiop:///" directs an EJB client to look for a name server on the local host listening on port 900. If not specified, this property defaults to the local host and port number 900, which is the same as specifying "iiop:///".

package

This property specifies the called program's package. The value is case sensitive. Specification of this property is required for run-time binding. If the package property is not specified, the default behavior is to use the package of the calling program.

This property is only applicable for calls to a Java program.

Valid parameter formats and linkage combinations by platform

The following tables show the linkage and run-time parameter formats valid for each type of run-time platform. The default format for the platform is marked in the table.

Table 53. Valid Parameters and Linkages for CICS Programs

Link Type	COMMPTR	COMMDATA	OSLINK	CICSOSLINK
DYNAMIC	Valid	Valid	Valid	Valid
STATIC	Valid	Valid	Valid	Valid
CICSLINK	Default	Valid		
REMOTE		Valid		
CSOCALL		Valid		

Table 54. Valid Parameters and Linkages for Non-CICS Host Programs

Link Type	COMMPTR	COMMDATA	OSLINK	CICSOSLINK
DYNAMIC			Default	
STATIC			Valid	
CICSLINK				
REMOTE				
CSOCALL				

Table 55. Valid Parameters and Linkages for GUI programs

Link Type	COMMPTR	COMMDATA	OSLINK	CICSOSLINK
DYNAMIC			Valid	
STATIC			Valid	
CICSLINK				
REMOTE	Valid	Valid		
CSOCALL	Valid	Valid		

Note: Default linkage for GUI programs is to call the program as a CMD or EXE file, passing parameter data in the command buffer separated by blanks. Modifications to the parameters by the called program are not returned to the caller.

Table 56. Valid Parameters and Linkages for C++ Programs

Link Type	COMMPTR	COMMDATA	OSLINK	CICSOSLINK
DYNAMIC			Default	
STATIC			Valid	
CICSLINK				
REMOTE	Valid	Valid		
CSOCALL	Valid	Valid		

Table 57. Valid Parameters and Linkages for Test Facility Calls to External Programs

Link Type	COMMPTR	COMMDATA	OSLINK	CICSOSLINK
DYNAMIC			Valid	
STATIC			Valid	
CICSLINK	Default for non-GUI programs	Valid		
REMOTE	Valid	Valid		
CSOCALL	Valid	Valid		

Note: Default linkage for test facility is determined at test time based on the following factors in the order listed:

- If the called program is defined in the library, the test facility interpretively executes the program out of the library
- Otherwise the program is called as a CMD or EXE file, passing parameter data in the command buffer separated by blanks. Modifications to the parameters by the called program are not returned to the caller.

Interfaces requiring a linkage table

A linkage table entry is not required for a called program if the default linkages for the run-time environment are acceptable. Table 58

Table 58. Nonstandard Linkages Supported by CALL

Function	Environment	Statement
Call to a PL/I program	MVS/VSE/VM Non-CICS	:calllink applname= <i>program-name</i> linktype=static
Call to a COBOL program or program that calls a PL/I program	MVS/VSE/VM Non-CICS	:calllink applname= <i>program-name</i> linktype=static
Static COBOL calls	CICS, MVS/VSE/VM Non-CICS	:calllink applname= <i>program-name</i> linktype=static ...

Table 58. Nonstandard Linkages Supported by CALL (continued)

Function	Environment	Statement
Dynamic COBOL calls	CICS	:calllink applname= <i>program-name</i> linktype=dynamic ...
Pass parameter values instead of pointers in the COMMAREA	CICS	:calllink applname= <i>program-name</i> parmform=COMMDATA ...
Call to a DLL in 16-bit mode	CICS for OS/2	:calllink applname= <i>program-name</i> linktype=dynamic
Call to a DLL in 16-bit mode	GUI program or test facility	:calllink applname= <i>program-name</i> linktype=dynamic bitmode=16 library= <i>value</i> ...
Call to a DLL (OS/2 or Windows) in 32-bit mode	GUI or C++ programs, or test facility	:calllink applname= <i>program-name</i> linktype=dynamic library= <i>value</i> ...
Call to a shared library (AIX, HP-UX, Solaris) in 32-bit mode	C++ programs	:calllink applname= <i>program-name</i> linktype=dynamic library= <i>value</i> ...
Call to a DLL with a program name longer than eight characters	C++ program	:calllink applname= <i>program-name</i> linktype=dynamic [externalname= <i>entry-point name</i>]...
Call a CICS for OS/2 program	Test facility	:calllink applname= <i>program-name</i> linktype=cicslink parmform=commptr [binform= <i>value</i>]
Call a CICS server program on a remote system	CICS	:calllink applname= <i>program-name</i> linktype=remote [contable= <i>value</i>] [location= <i>value</i>] [luwcontrol= <i>value</i>] [serverid= <i>value</i>]

Table 58. Nonstandard Linkages Supported by CALL (continued)

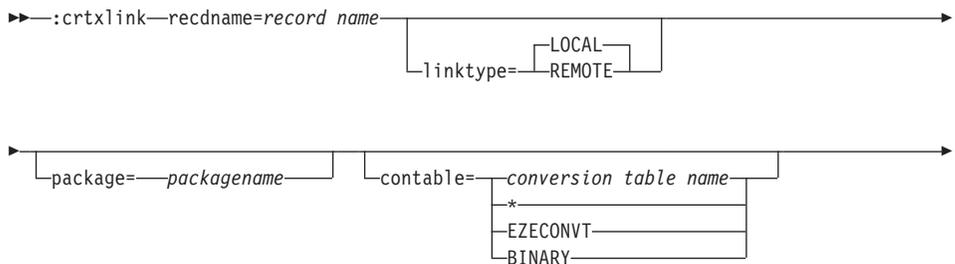
Function	Environment	Statement
Call a server program on a remote system	GUI program, OS/2, AIX, Windows NT, or Test facility	:calllink applname= <i>program-name</i> linktype=remote [externalname= <i>value</i>] [library= <i>value</i>] [luwcontrol= <i>value</i>] [remoteapptype= <i>value</i>] [remotecomtype= <i>value</i>] [serverid= <i>value</i>]
Call a non-VisualAge Generator MVS or VSE CICS program	Test facility	:calllink applname= <i>program-name</i> linktype=ciclink parmform=commdata contable= <i>value</i>
Call any program from a Java client	Needed when calls are made from one platform to another. It is recommended in all cases.	contable=CSOpxxx

Specifying CREATX linkage (CRTXLINK)

The CRTXLINK tag specifies the type of linkage being generated for CALL CREATX in the CICS environment. Specify the CRTXLINK tag only if the created transaction is to be started on a remote CICS system. The defaults are acceptable for transactions started on the same system.

For Java programs, CREATX linkage must be specified if the calling program uses CREATX to start a program in a different package.

The following diagram shows the attributes you can specify for the CRTXLINK tag:





Definitions for CRTXLINK

recdname

Specifies the name of the CREATX record on the CALL CREATX statement for the linkage being defined.

You can use an asterisk (*) as a global substitution character in the record name parameter; however, it is only valid as the last or only character.

linktype

Specifies the type of linkage being generated.

LOCAL Specifies that the transaction being created is on the same system as the program containing the CREATX statement.

This value is the default value.

REMOTE Specifies that the transaction being created might be on a different system than the program containing the CREATX statement.

package

Specifies the name of the package that contains the program to be run. The default is the package of the program that contains the CALL CREATX statement.

This property is only valid for Java programs.

contable

Specifies the name of the conversion table used to perform automatic data conversion on a remote CREATX statement.

This attribute value is used only if the linkage type is specified as REMOTE.

Data conversion is specified as follows:

- If the CONTABLE attribute is not specified, data conversion is not performed automatically by the program when the record is used in a CREATX statement. In this case, the program must perform any required conversion either by using explicit calls to the EZECONV special function word or by defining a conversion template for the operation to CICS.
- If the CONTABLE attribute is specified, you can specify the following values:

conversion table name

Conversion is performed on the client using the conversion table specified.

EZECONVT

Conversion is performed on the client using the conversion table name in the EZECONVT special function word at run time. If EZECONVT contains blanks, no conversion is performed.

Some conversion table names have special meaning:

- * Conversion is performed on the client using the default conversion table.

On OS/2, AIX, and Windows systems the default is the conversion table specified in environment variable EZERCVT. If EZERCVT is not specified, the default is conversion table ELAxxxxx (OS/2 or AIX) or ELACWxxx (Windows) where xxxxx or xxx is the code specified in environment variable EZERNLS. If EZERNLS is not specified, the default national language code is ENU.

On MVS or VSE systems, the default conversion table is ELAxxxxx where xxxxx is the code specified when the calling program was generated (TARGNLS generation option).

BINARY

Only binary fields are converted. The byte order in the binary field is reversed.

This table is used with OS/2 and Windows clients communicating with AIX servers, and vice versa, when both the client and the server are running under the same code page.

Refer to the system and program guide for CICS for OS/2 for information on defining a conversion template to CICS.

location

Specifies how the location (CICS system identifier) of a remote transaction is defined.

This attribute value is used only if the linkage type is specified as REMOTE.

CICS Specifies that the location for the remote transaction named in the CREATX record at run time is defined in the program control table (PCT)

EZELOC

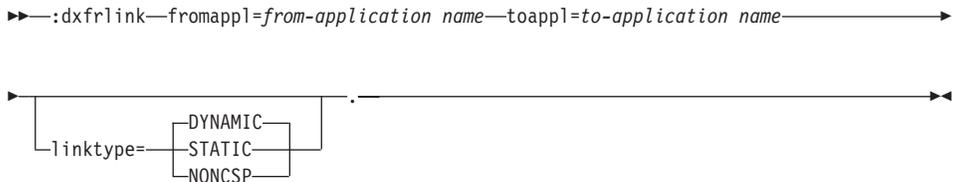
Specifies that the location for the remote transaction is obtained from the EZELOC special function word when the CREATX operation is performed.

The location is passed to CICS using the SYSID keyword on the EXEC CICS command.

Specifying DXFR linkage (DXFRLINK)

The DXFRLINK tag specifies the type of linkage to be generated when a main program uses a DXFR statement to pass control to other programs.

The following diagram shows the attributes you can specify for the DXFRLINK tag:



Note: The DXFRLINK tag is supported for host and CICS for OS/2 programs.

Definitions for DXFRLINK

fromappl

Specifies the name of the program that transfers control using a DXFR statement.

For non-host programs, this is the first main program in the run unit; the program is not necessarily the program that contains the DXFR statement.

For CICS programs, the from program is always the program that contains the DXFR statement.

The DXFRLINK tag does not support using the asterisk (*) for global character substitution.

toappl Specifies the VisualAge Generator program or non-VisualAge Generator program that is the target of a DXFR statement.

For non-CICS host programs, the to program includes all programs started by a DXFR statement from the initial main program, and all programs that the transferred-to programs transfer to using a DXFR statement. This also includes non-VisualAge Generator programs being transferred to with a DXFR statement.

For CICS programs, the to program is the name of the non-VisualAge Generator program being transferred-to.

If the EZEAPP special function word is specified as the target program on the DXFR statement, do not specify EZEAPP as the to program on the DXFRLINK linkage table entry. Instead, specify the name of the program that will be in EZEAPP when the from program runs.

For example, in the MVS/TSO environment, Program A transfers program control to program B using a DXFR statement. Program B, in turn, transfers control to program C using a DXFR statement. Program C transfers control (using a DXFR statement) to the program specified by the EZEAPP special function word, where EZEAPP is set to program D or program E. You would have to specify the following programs in the linkage table:

```
fromappl=A  toappl=B  
fromappl=A  toappl=C  
fromappl=A  toappl=D  
fromappl=A  toappl=E
```

For CICS environments, with the same set of programs, a linkage table is not required. However, if D is a non-VisualAge Generator program, the linkage table required for CICS is as follows:

```
fromappl=C  toappl=D  linktype=noncsp
```

linktype

Specifies the type of linkage being generated.

DYNAMIC

For non-CICS host programs, specifies that a dynamic COBOL call is to be generated in the initial main program for a DXFR operation to a generated program

For CICS programs, an XCTL is used to implement the DXFR statement. The target program is assumed to be a generated program, and run-unit status information is passed to the target program in addition to the DXFR parameter record.

This is the default value.

STATIC

For non-CICS host programs, specifies that a static COBOL call is to be generated in the main program for a DXFR operation to a non-CICS generated program

For CICS programs, this option is treated the same as the DYNAMIC option.

For non-CICS environments, this value is required for target programs that call PL/I programs or that call programs that call PL/I programs.

NONCSP

For MVS, VM, and CICS programs, specifies that an XCTL is to be used to implement the DXFR statement.

DXFR to a non-VisualAge Generator program is not supported in the VSE batch environment.

All resources allocated by Server for MVS, VSE, and VM or VisualAge Generator Server are released.

The NONCSP value is required for DXFR statements to non-VisualAge Generator programs. The NONCSP value can be specified either as an option on the DXFR statement or in the linkage table.

This value is the only value that is valid for CICS programs.

Interfaces requiring a linkage table

A linkage table entry is not required for generation of a DXFR between programs if the default linkages for the run-time environment are acceptable. Table 59 statements, and which linkage table tags and keywords are needed to control each linkage.

Table 59. Nonstandard Linkages Supported For DXFR

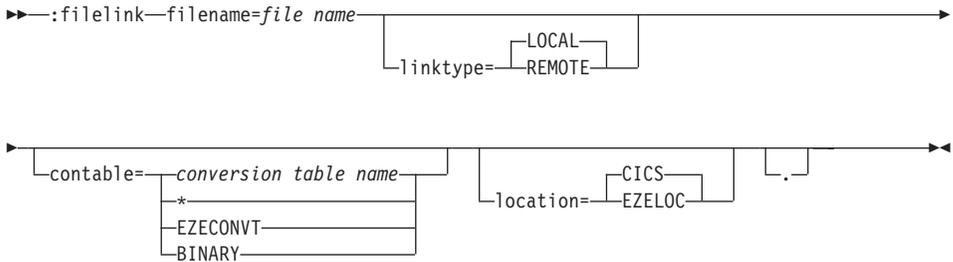
Function	Environment	Statement
Transfer using a DXFR to a generated program that calls a PL/I program	MVS/VSE/VM Non-CICS	:dxfrlink fromappl= <i>fromhyphen.program-name</i> toappl= <i>to-program-name</i> linktype=static
Transfer using a DXFR statement to a non-VisualAge Generator program, where the NONCSP option was not specified on the DXFR statement	CICS, MVS, and VM programs	:dxfrlink fromappl= <i>from-program-name</i> toappl= <i>to-program-name</i> linktype=noncsp

Specifying File linkage (FILELINK)

The FILELINK tag specifies the type of file access generated for CICS-managed VSAM files and transient data queues. Specify the FILELINK tag only if a file is remote; the default file access is acceptable for local files.

Linkage table file entries are accessed only for files specified as CICS VSAM files and transient data queues. Other types of files are always accessed as local files.

The following diagram shows the attributes you can specify for the FILELINK tag:



Definitions for FILELINK

filename

Specifies the VisualAge Generator file name for the file access being defined.

You can use an asterisk (*) as a global substitution character in the file name parameter; however, it is only valid as the last or only character.

linktype

Specifies the type of linkage being generated.

LOCAL Specifies that the file resides on the same system as the program.

This is the default value.

REMOTE Specifies that the file might reside on a different system than the program.

This value is valid only for CICS-managed VSAM files and transient data queues.

contable

Specifies the name of the conversion table used to perform automatic data conversion for remote file input/output access.

This attribute value is used only if the linkage type is specified as REMOTE.

Data conversion is specified as follows:

- If the CONTABLE attribute is not specified, data conversion is not performed automatically by the program for the file input/output. In this case, the program must perform any required conversion either by using explicit calls to the EZECONV special function word or by defining a conversion template for the operation to CICS.

- If the **CONTABLE** attribute is specified, you can specify the following values:

conversion table name

Conversion is performed on the client using the conversion table specified.

EZECONVT

Conversion is performed on the client using the conversion table name in the **EZECONVT** special function word at run time. If **EZECONVT** contains blanks, no conversion is performed. **EZECONVT** is not supported in calls from GUI client programs.

Some conversion table names have special meaning:

- * Conversion is performed on the client using the default conversion table.

On OS/2, AIX, and Windows systems the default is the conversion table specified in environment variable **EZERCVT**. If **EZERCVT** is not specified, the default is conversion table **ELAxxxxx** (OS/2 or AIX) or **ELACWxxx** (Windows) where **xxxxx** or **xxx** is the code specified in environment variable **EZERNLS**. If **EZERNLS** is not specified, the default national language code is **ENU**.

On MVS or VSE systems, the default conversion table is **ELAxxxxx** where **xxxxx** is the code specified when the calling program was generated (/TARGNLS generation option).

BINARY

Only binary fields are converted. The byte order in the binary field is reversed.

This table is used with OS/2 and Windows clients communicating with AIX servers, and vice versa, when both the client and the server are running under the same code page.

Refer to the system and program guide for CICS for OS/2 for information on defining a conversion template to CICS.

location

Specifies how the location (CICS system identifier) of a remote file is defined.

This attribute is ignored if the linkage type is not **REMOTE**.

CICS Specifies that the location for a remote file is defined in the CICS file control table (FCT) or destination control table (DCT).

EZELOC

Specifies that the location for the remote file is obtained from the EZELOC special function word when an input or output operation is performed to the file.

The location is passed to CICS using the SYSID keyword on the EXEC CICS command.

Sample linkage table entries

The following are examples of linkage table entries:

- APP1 is an existing non-VisualAge Generator CICS COBOL program expecting data to be passed in the COMMAREA:
:calllink applname=app1 parmform=commdata linktype=cicslink.
- APP2 is an MVS/TSO or VM CMS program that uses a DXFR statement to transfer control to APP3, which uses a DXFR statement to return control to APP2. Because APP2 is the initial program, it must be identified as the FROMAPPL. Because APP3 is never an initial program in a run unit, it does not have to be identified as a FROMAPPL. Both APP2 and APP3 call PL/I programs. All the PL/I program names begin with the string P1.
:calllink applname=p1* linktype=static parmform=oslink.
:dxfrlink fromappl=app2 toappl=app3 linktype=static.

A DXFRLINK linkage table entry is not required for the DXFR back to APP2 because a DXFR back to the initial program is handled in the internal logic flow of the initial program.

- The installation chooses CICSOSLINK as the default linkage for CICS VisualAge Generator programs. The non-VisualAge Generator program named NCPGM does not contain any CICS functions and expects conventional COBOL linkage to be used.
:calllink applname=nccpgm linktype=dynamic parmform=oslink.
:calllink applname=* linktype=dynamic parmform=cicsoslink.
- This linkage table entry would be used in a call from a Java GUI client to server program APP1 that is installed on CICS Region NRACICS2 using conversion table CSOE037.
:calllink applname=APP1 linktype=CSOCALL remotecomtype=CICSCLIENT
location=NRACICS2 contable=CSOE037.

The following example is a linkage table entry for local calls between two VAGen Java server programs.

```
:calllink applname=SERVER package='my.pkg' linktype=DYNAMIC
```

Note: This linkage must be supplied at generation time. The called server (SERVER) program's package name must be quoted because it contains a dot. If no linktype is specified, the value DYNAMIC is used by default.

Chapter 26. Link edit parts

Linkage Editor control statement files are used for the MVS, VSE, and VM environments. Preparation templates shipped with VisualAge Generator Developer contain default linkage editor control statements for generated programs. If a program calls or is called by other programs using static COBOL calls, you must define additional linkage editor control statements in a program-specific part.

Link-editing static COBOL calls

Define a link edit part when any of the following is true:

- The program includes static COBOL calls to other programs.
- The program is called by other programs using a static COBOL call.

A static COBOL call requires that the called and calling programs be link-edited into the same load module.

You can use a dynamic COBOL call or CICS link instead of a static COBOL call to avoid the extra link-editing. However, when programs contain CALL and DXFR statements to specific programs, you must generate the static COBOL calls.

If static calls are used in generated programs, the following statements are true:

- Linkage editor control statements must be defined for each program that uses static linkage to call another program or to transfer control to another program.
- Linkage editor control statements must be defined for each program that is called by another program or is transferred to by another program.

Defining a link edit part

Linkage Editor control statements for a program must be defined in a link edit part. The link edit part must be named *prgmname.linkedit_value* if you use the /LINKEDIT generation option to generate the program or *prgmname* if you do not use /LINKEDIT. We recommend that you always use the /LINKEDIT option, because it is difficult to create a link edit part with the same name as the program part. You must use the /LINKEDIT generation option if you need to define more than one link edit part for the same program. This may be because you generate the same program for more than one target system.

It is best to define the program's link edit part in the same package/application as the program so that it is available whenever the program is available. It must be contained in the configuration map used to generate the program. The link edit part must contain linkage editor control statements for linking every load module that contains the program. See “/LINKEDIT (Link edit)” on page 326 for more information on specifying the linkage editor part to be used when preparing the code generated for the program.

The control statements in the link edit part must follow linkage editor syntax and format rules:

- Statements cannot start in column one.
- No blank lines are allowed between statements.

Refer to the linkage editor document for your host system for more information about using linkage editor control files.

Linkage Editor control statements for MVS environments

When VisualAge Generator Developer produces preparation JCL for MVS environments, it queries for the existence of the link edit part. If the part exists, an additional link-edit step is generated for the program using the template EFK2MPRE. The generated JCL calls the cataloged procedure ELARLINK. The part containing the linkage editor control statements is included in the generated JCL as the linkage editor SYSLIN input file for the link-edit step.

MVS program with static calls to other programs

A link edit part containing linkage editor control statements must be defined for each generated program that contains static COBOL calls. The linkage editor control statements consist of ENTRY and NAME statements for the load module and INCLUDE statements for each statically called program and the base program. The VisualAge Generator Server for MVS, VSE, and VM entry stub program (ELARMAIN) must also be included in the MVS/TSO, MVS batch, and IMS BMP environments. Other VisualAge Generator Server for MVS, VSE, and VM and database stubs are included along with the programs from the original link-edit step produced during the generation process.

Table 60 on page 247 shows sample control statements by environment for a specific example where program BASEPGM statically calls both a generated COBOL program VGNAP1 and a PL/I program PLIAPP1. In the example, the object decks for PL/I programs have been written to data set NONVGN.OBJ.LIBRARY. All the INCLUDE statements for the called members precede the INCLUDE for the calling program.

Table 60. Sample control statements

Environment	Base program type	Control statements in BASEPGM
CICS for MVS/ESA IMS/VS	Main or called	INCLUDE SYSLMOD(VGNAP1) INCLUDE NONVGNL(PLIAPP1) * INCLUDE SYSLMOD(BASEPGM) ENTRY BASEPGM NAME BASEPGM(R)
MVS/TSO MVS batch IMS BMP	Main	CHANGE ELAAPPL(BASEPGM) INCLUDE SELALMD(ELARMAIN) INCLUDE SYSLMOD(VGNAP1) INCLUDE NONVGNL(PLIAPP1) INCLUDE SYSLMOD(BASEPGM) ENTRY ELARMAIN NAME BASEPGM(R)
MVS/TSO MVS batch IMS BMP	Called	INCLUDE SYSLMOD(VGNAP1) INCLUDE NONVGNL(PLIAPP1) INCLUDE SYSLMOD(BASEPGM) ENTRY BASEPGM NAME BASEPGM(R)

*PL/I programs cannot be statically linked for CICS for MVS/ESA. For CICS for MVS/ESA, this link would be to a statically linked non-VisualAge Generator COBOL program.

Figure 68 shows how a project leader has customized the ELARLINK procedure by adding the following DD statement to allocate the data set.

```
//NONVGNL DD DSN=NONVGN.OBJ.LIBRARY,DISP=SHR
```

Figure 68. Customized ELARLINK procedure

MVS programs that are statically called by other programs

A link edit part that contains link-edit statements must also be defined for each program that is the target of a static COBOL call from another program. Include statements to link every load module that calls a program in the link edit part so that each load module is automatically linked again whenever the called program is generated.

To define the link edit part containing the linkage editor control statements, add the control statements defined for the base program to the link edit part that contains link-edit statements defined for each included program.

See “MVS program with static calls to other programs” on page 246 for an example showing the linkage editor control statements defined for the BASEPGM program being added to the link edit part containing link-edit statements for the VGNAP1 program. When the VGNAP1 program is

generated, the BASEPGM program is automatically linked again to include the new version of the VGNAP1 program.

Modifying the ELARLINK Procedure

If a program is both the target of a static COBOL call and also includes a static COBOL call to other programs, you must further tailor the ELARLINK procedure to use different load libraries as the include and output libraries for generated program load modules.

Figure 69 shows a SYSLMOD DD statement before modifications.

```
//SYSLMOD DD DISP=SHR,DSN=&CGHLQ..&ENV..LOAD
```

Figure 69. SYSLMOD DD statement

Figure 70 shows the replacement DD statements for the SYSLMOD DD statement.

```
//VGNINCL DD DISP=SHR,DSN=&CGHLQ..&ENV..LOAD  
//SYSLMOD DD DISP=SHR,DSN=&CGHLQ..&ENV..RELINK.LOAD
```

Figure 70. SYSLMOD DD statement replacements

Then define all link edit parts containing link-edit statements to use the VGNINCL library as the include library for generated programs. Setting up the libraries this way has the following effects:

- The initial link-edit step in the preparation procedures links the generated program with VisualAge Generator Server for MVS, VSE, and VM and database stub programs into the VGNINCL library.
- Statically called programs are not linked together in the VGNINCL library to avoid the same program being included multiple times on the relink step.
- The relink step combines all statically linked programs into the appropriate load modules and stores the load modules in the SYSLMOD library.

Specifying the control statements

Table 61 on page 249 shows how the control statements for linking program BASEPGM would be specified if statically called program VGNAP1 in turn statically called program VGNAP2. The control statements for linking program BASEPGM would also be added to the VGNAP1 and VGNAP2 link edit parts so that program BASEPGM is linked again when either program VGNAP1 or VGNAP2 is generated.

Table 61. Control statements example

Environment	Base program type	Control statements in BASEPGM
CICS for MVS/ESA IMS/VS	Main or called	INCLUDE VGNINCL(VGNAP1) INCLUDE VGNINCL(VGNAP2) INCLUDE NONVGNL(PLIAPP1) * INCLUDE VGNINCL(BASEPGM) ENTRY BASEPGM NAME BASEPGM(R)
MVS/TSO MVS batch IMS BMP	Main	CHANGE ELAAPPL(BASEPGM) INCLUDE SELALMD(ELARMAIN) INCLUDE VGNINCL(VGNAP1) INCLUDE VGNINCL(VGNAP2) INCLUDE NONVGNL(PLIAPP1) INCLUDE VGNINCL(BASEPGM) ENTRY ELARMAIN NAME BASEPGM(R)
MVS/TSO MVS batch IMS BMP	Called	INCLUDE VGNINCL(VGNAP1) INCLUDE VGNINCL(VGNAP2) INCLUDE NONVGNL(PLIAPP1) INCLUDE VGNINCL(BASEPGM) ENTRY BASEPGM NAME BASEPGM(R)

*PL/I programs cannot be statically linked for CICS for MVS/ESA. For CICS for MVS/ESA, this link would be to a statically linked non-VisualAge Generator COBOL program.

For CICS for MVS/ESA, MVS batch, IMS/VS, and IMS BMP environments, the relink load library and the original load libraries are required when running programs. The relink library is required for statically linked modules. The original library is required for modules that did not have to be statically linked. Figure 71 shows how to customize the run-time JCL file and the IMS and CICS for MVS/ESA region JCL file to include the load libraries in the correct order in the STEPLIB statement.

```
// DD DISP=SHR,DSN=cgh1q.env.RELINK.LOAD
// DD DISP=SHR,DSN=cgh1q.env.LOAD
```

Figure 71. Load libraries on the STEPLIB statement

The *chglq* and *env* are the values that were substituted for the CGHLQ and ENV parameters in the EFK2MPRE procedure.

For the MVS/TSO environment, the library that runs the first load module in a run unit is the library that is searched first for any other modules that are

required. Figure 72 shows a sample library where you can put all the modules.

```
cgh1q.env.RELINK.LOAD
```

Figure 72. Modules in a single library

Change the run-time CLIST templates to use `cgh1q.env.RELINK.LOAD`. In addition, add link edit parts for all programs that are linked stand-alone so that they are moved into `cgh1q.env.RELINK.LOAD`. Table 62 shows the link-edit commands required for a program called `ALONE` in the MVS/TSO environment.

Table 62. Sample link-edit statements for a stand-alone program

Base program type	Control statements
Main	CHANGE ELAAPPL(ALONE) INCLUDE SELALMD(ELARMAIN) INCLUDE VGNINCL(ALONE) ENTRY ELARMAIN NAME ALONE(R)
Called	INCLUDE VGNINCL(ALONE) ENTRY ALONE NAME ALONE(R)

Linkage Editor control statements for VM environments

When the VisualAge Generator Developer produces the preparation REXX for VM environments, it queries for the existence of the link edit part. If the part exists, the linkage editor control statements in the part are used to generate the linkage edit control statement file.

VM programs with static calls to other programs

A link edit part with linkage editor control statements must be defined for each generated program that contains static COBOL calls. The linkage editor control statements consist of `ENTRY` and `NAME` statements for the load module and `INCLUDE` statements for each statically called program and the base program. The VisualAge Generator Server for MVS, VSE, and VM entry stub program (`ELARMAIN`) must also be included in the VM CMS and VM batch environments. Other VisualAge Generator Server for MVS, VSE, and VM and database stubs are included along with the programs from the original link-edit step produced during the generation process.

Table 63 on page 251 shows sample control statements by environment for a specific example where program `BASEPGM` statically calls both a generated COBOL program `VGNAP1` and a PL/I program `PLIAPP1`. In the example, the object decks for PL/I programs have been written to a CMS minidisk, and

ELAPREP EXEC has been modified to issue a FILEDEF to define this file as data set PLIAPP1. All the INCLUDE statements for the called members precede the INCLUDE for the calling program.

Table 63. Sample control statements

Environment	Base program type	Control statements in BASEPGM
VM CMS VM batch	Main	CHANGE ELAAPPL(BASEPGM) INCLUDE SELALMD(ELARMAIN) INCLUDE SYSLMOD(VGNAP1) INCLUDE NONVGNL(PLIAPP1) INCLUDE SYSLMOD(BASEPGM) ENTRY ELARMAIN NAME BASEPGM(R)
VM CMS VM batch	Called	INCLUDE SYSLMOD(VGNAP1) INCLUDE NONVGNL(PLIAPP1) INCLUDE SYSLMOD(BASEPGM) ENTRY BASEPGM NAME BASEPGM(R)

A second method to achieve the same effect is to use local TXTLIBs to hold the object decks of non-VisualAge Generator programs that are called by a VisualAge Generator program. Using the example from Table 63, the object decks for PLIAPP1 are built into a local TXTLIB called MYTXTLIB, and the BASEPGM EXECF file is modified to pass the name of the local TXTLIB to ELAPREP as the USERLKED parameter (USERLKED=MYTXTLIB). This method is easier than the first method when more than one object deck for the called program or multiple called programs are involved.

VM programs that are statically called by other programs

A link edit part that contains link-edit statements must also be defined for each program that is the target of a static COBOL call from another program. Include statements to link every load module that calls a program in the link edit part so that each load module is automatically linked again whenever the called program is generated.

To define the link edit part containing the linkage editor control statements, add the control statements defined for the base program to the link edit part that contains link-edit statements defined for each included program.

See Table 63 for an example showing the linkage editor control statements defined for the BASEPGM program being added to the link edit part containing link-edit statements for the VGNAP1 program. When the VGNAP1 program is generated, the BASEPGM program is automatically linked again to include the new version of the VGNAP1 program.

Specifying the control statements

Table 64 shows how the control statements for linking program BASEPGM would be specified if statically called program VGNAP1 in turn statically called program VGNAP2. The control statements for linking program BASEPGM would also be added to the VGNAP1 and VGNAP2; link edit parts so that program BASEPGM is linked again when either program VGNAP1 or VGNAP2 is generated.

Table 64. Control statements example

Environment	Base program type	Control statements in BASEPGM
VM CMS VM batch	Main	INCLUDE TXTDECK CHANGE ELAAPPL(BASEPGM) INCLUDE SELALMD(ELARMAIN) INCLUDE SELALMD (ELARSINT) INCLUDE PLIAPP1 INCLUDE SYSLMOD (BASEPGM) INCLUDE ELAEUOPT INCLUDE ELABXITB ENTRY ELARMAIN NAME BASEAPP
VM CMS VM batch	Called	INCLUDE TXTDECK INCLUDE SELALMD (ELARSINT) INCLUDE PLIAPP1 INCLUDE SYSLMOD (BASEPGM) INCLUDE ELAEUOPT INCLUDE ELABXITB ENTRY BASEAPP NAME BASEAPP

Note: Add these control statements if any of the programs use SQL statements:
INCLUDE SQLSTUB
INCLUDE SQLCOBOL

For VM batch environments, the relink load library and the original load libraries are required when running programs. The relink library is required for statically linked modules. The original library is required for modules that did not have to be statically linked.

For the VM CMS environment, the library that runs the first load module in a run unit is the library that is searched first for any other modules that are required. Figure 73 shows a sample library where you can put all the modules.

```
cgh1q.env.RELINK.LOAD
```

Figure 73. Modules in a single library

Change the run-time CLIST templates to use `cghlq.env.RELINK.LOAD`. In addition, add link edit parts for all programs that are linked stand-alone so that they are moved into `cghlq.env.RELINK.LOAD`. Table 65 shows the link-edit commands required for a program called `ALONE` in the VM CMS environment.

Table 65. Sample link-edit statements for a stand-alone program

Base program type	Control statements
Main	CHANGE ELAAPPL(ALONE) INCLUDE SELALMD(ELARMAIN) INCLUDE VGNINCL(ALONE) ENTRY ELARMAIN NAME ALONE(R)
Called	INCLUDE VGNINCL(ALONE) ENTRY ALONE NAME ALONE(R)

Linkage Editor control statements for VSE environments

When VisualAge Generator Developer produces preparation JCL for VSE environments, it queries the existence of the link edit part. If the part exists, it is used for the link-edit step of the preparation job. If static calls are used for VSE, linking for the static calls occurs during the regular link-edit step. You can use the default link-edit templates and modify them as needed to add static modules.

See “Program templates for CICS for VSE/ESA and VSE batch preparation” on page 64 for a list of link-edit templates.

VSE program with static calls to other programs

Table 66 on page 254 shows sample control statements by environment for a specific example where program `BASEPGM` statically calls both a generated COBOL program `VGNAP1` and a PL/I program `PLIAPP1`. In the example, the object decks for the `PLIAPP1` and `VGNAP1` are stored in the sublibrary `PRD5.NONVGNLIB`.

Figure 74 shows the statement the system administrator must set for the value for the `VUSERLIB` symbolic parameter in the generation options file.

```
/SYMPARM=VUSERLIB, 'PRD5.NONVGNBL'
```

Figure 74. Setting VUSERLIB

All the `INCLUDE` statements for the called programs precede the `INCLUDE` for the calling program.

Table 66. VSE Linkage Editor control statements showing static calls to programs

Environment	Base program type	Control statements
CICS for VSE/ESA	Main or Called	PHASE BASEPGM,* INCLUDE DFHECI INCLUDE ELARSINC INCLUDE ELARVCS INCLUDE ELAASSGN INCLUDE ELAASADR INCLUDE VGNAP1 INCLUDE PLIAPP1 INCLUDE DFHPL1I INCLUDE BASEPGM ENTRY BASEPGM
VSE batch	Main or Called	PHASE BASEPGM,* INCLUDE ELARSINV INCLUDE ELARVCS INCLUDE ELAASSGN INCLUDE ELAASADR INCLUDE VGNAP1 INCLUDE PLIAPP1 INCLUDE BASEPGM ENTRY BASEPGM

*Static link to a PL/I program. PL/I programs cannot be statically linked for CICS for VSE/ESA. For CICS for VSE/ESA, this link would be to a statically linked non-VisualAge Generator COBOL program.

When the VGNAP1 program is generated, the BASEPGM program is automatically linked again to include the new version of the VGNAP1 program.

The control statements in the link edit part must follow linkage editor syntax and format rules:

- Statements cannot start in column one.
- Statements cannot have blank lines between them.

Refer to the linkage editor document for your host system for more information about linkage editor control statements.

VSE programs that are statically called by other programs

A link edit part that contains link-edit statements must also be defined for each program that is the target of a static COBOL call from another program. Include statements to link every load module that calls a program in the link edit part so that each load module is automatically linked again whenever the called program is generated.

To define the link edit part containing the linkage editor control statements, add the control statements defined for the base program to the link edit part that contains link-edit statements defined for each included program.

VSE programs with static links to and from other programs

Table 67 shows how the control statements for linking program BASEPGM would be specified if statically called program VGNAP1 in turn statically linked to program VGNAP2. The control statements for linking program BASEPGM would also be added to the VGNAP1 and VGNAP2 link edit parts so that program BASEPGM is relinked when either program VGNAP1 or VGNAP2 is generated.

Table 67. VSE Linkage Editor control statements showing static calls to and from programs

Environment	Base program type	Control statements in BASEPGM
CICS for VSE/ESA	Main or called	PHASE BASEPGM,* INCLUDE DFHECI INCLUDE ELARSINC INCLUDE ELARSVCS INCLUDE ELAASADR INCLUDE VGNAPI INCLUDE VGNAP2 INCLUDE PLIAPP1 INCLUDE DFHPLI1 INCLUDE BASEPGM ENTRY BASEPGM
VSE batch	Main or called	PHASE BASEPGM,* INCLUDE ELARSINV INCLUDE ELARSVCS INCLUDE ELAASSGN INCLUDE ELAASADR INCLUDE VGNAP1 INCLUDE VGNAP2 INCLUDE PLIAPP2 INCLUDE BASEPGM ENTRY BASEPGM

*Static link to a PL/I program. PL/I programs cannot be statically linked for CICS for VSE/ESA. For CICS for VSE/ESA, this link would be to a statically linked non-VisualAge Generator COBOL program.

Specifying AMODE and RMODE

Most generated programs must be linked AMODE(31) and RMODE(ANY). You can link-edit generated programs as AMODE(24) and RMODE(24) only if the generated program must be statically linked with a non-VisualAge Generator program that must run AMODE(24). When AMODE(24) is required,

you can add user-defined symbols for AMODE and RMODE in preparation JCL templates. When AMODE(24), AMODE(ANY), or RMODE(24) are required for a VM program, you can specify the AMODE, RMODE, or both on the call to ELAPREP. See “Modifying preparation templates and EXECs for VM environments” on page 97 for information on specifying parameters for the ELAPREP EXEC.

See the Chapter 24. Generation options parts for information on the DATA compiler option.

Error return codes on static links

- 8 When an error occurs during the initial MVS link-edit step, the return code specifies that there are unresolved external references for statically called programs.

 The problem is resolved on the relink step.
- 8 When an error occurs during the initial MVS relink step or during the VSE or VM link-edit step, the return code specifies that there are unresolved references.

 Some of the included programs were not generated and prepared.
- 12 When an error occurs during the MVS relink step or during the VSE or VM link-edit step, the return code indicates a missing module and entry point in the base program.

Do not use a generated program until all programs that it runs with are successfully prepared and all load modules are linked with a completion code of 0.

Chapter 27. BIND control parts

A BIND control part is required only for DB2 programs generated for MVS systems. Preparation templates shipped with VisualAge Generator contain default BIND command statements used to generate a default BIND control part when a DB2 program is generated. The template used to produce the BIND control part is determined by the target environment, whether a DB2 work database is used, and whether a DL/I database is used. For a list of templates used, see “Preparation JCL for MVS or VSE” on page 62. When the default BIND control part is used during preparation, a DB2 plan is produced that has the same name as the program, and includes only the program database request module (DBRM) and the VisualAge Generator Server for MVS, VSE, and VM as members. DBRMs are needed to gain access to the DB2 work database or to issue SQL COMMIT WORK or ROLLBACK WORK commands if required. The default plan is suitable when the initial program is the only program that needs to access the SQL tables in the run unit.

If the default BIND control part generated is not suitable, it is necessary to provide a BIND control part as an input to generation. See “Considerations for plan definition” on page 258 for more details about when a default BIND control part might not be suitable.

To determine whether a BIND control part is to be produced from default templates or from a BIND control part, the generator searches the Smalltalk image into which the generation configuration map was loaded or the Java workspace into which the generation projects were loaded. The generator looks for a BIND control part named *prgmname.bind_value* if you specified the /BIND option or *prgmname* if you did not specify /BIND. If a matching BIND control part is found, it is used to produce the program’s BIND control part. Otherwise, the program’s BIND control part is produced from the appropriate template. Regardless of the input used to produce the BIND control part, the BIND command file is named *prgmname.BDC*.

We recommend that you always use the /BIND option, because it is difficult to create a bind control part with the same name as the program part.

Defining BIND control parts

The project leader must determine the DB2 plans that are needed for a project and define the BIND control parts for plans that include multiple DBRMs. The program names specified on the MEMBER keyword on the BIND command identify the DBRMs to be included in the plan. When the BIND command for a plan includes more than one generated program, that command must be

specified in BIND control parts for each of the generated programs. The BIND control part for each program must be named `prgname` if the `/BIND` generation option is not used to generate the program, or `prgname.bind_value` if the `/BIND` generation option is used. `bind_value` may be any character string, but a good convention would be to identify why you need to differentiate between two BIND control parts for the same program. For example, if you need different BIND control parts for generating the same program for CICS for MVS/ESA and IMS/VS target systems, you might name the required BIND control parts `prgname.MVS` and `prgname.IMS`.

It is best to define the program's BIND control part in the same package/application as the program so that it is available whenever the program is available. It at least has to be contained in the configuration map or projects used to generate the program.

If a generated program is included in more than one plan, its BIND control part must contain the BIND command for each plan that it is a part of. Whenever the program is generated and prepared, the BIND step of preparation binds every plan that includes that program.

See "Chapter 29. Generation command and option descriptions" on page 297 for more information on specifying the BIND control part to be used when generating the program.

Considerations for plan definition

Consider several issues as you define BIND commands for a program. These issues include the following:

- The program and plan names
- The use of XFER, DXFR, CALL and CONVERSE statements, and the `/RT` generation option
- The use of host services database request modules (DBRMs)

Naming CICS for MVS/ESA program plans

For CICS for MVS/ESA programs, any plan name can be used. The plan must be associated with the CICS transaction in the CICS resource control table (RCT). More than one CICS transaction code can be associated with a single DB2 plan name in the RCT.

Naming MVS/TSO and MVS batch program plans

For MVS/TSO and MVS batch programs, the plan used at run time is specified on the DSN subcommand RUN used to start the initial program. The sample run-time JCL and CLISTs built by the VisualAge Generator Developer assume the plan name is the same as the initial program name. You can modify the plan name before the JCL or CLIST is used.

Naming IMS program plans

For IMS programs, the name of the program plan must match the name of the initial program load module for the transaction or batch job. Do not modify the name of the plan.

Effects of XFER, DXFR, CALL, CONVERSE, and the /RT generation option on plans

For CICS for MVS/ESA and IMS programs, if a program uses an XFER statement or the /RT generation option to transfer to another program, the DBRMs for these programs do not have to be bound together. Transferring to another program using an XFER statement or with the /RT generation option results in a transaction switch. For IMS transactions, a separate plan must be used because the plan name must match the name of the program associated with the new transaction. CICS for MVS/ESA enables you to use the same plan name for both transactions. However, when transferring with an XFER statement or with the /RT generation option, there is reason to bind the programs into a single plan.

If a program does a segmented CONVERSE in IMS or CICS environments and switches transaction identifiers, the program database request module (DBRM) must be bound into the plan of each transaction in which the program issues an SQL request.

If a program uses an XFER statement or the /RT generation option in the MVS/TSO, MVS batch, and IMS BMP environments, the DBRMs from each program must be bound into the same plan.

A CALL or DXFR statement does not result in a transaction switch. The DBRMs of all programs and non-VisualAge Generator programs connected by CALL or DXFR statements must be bound into the same plan.

Using host services CICS for MVS/ESA DBRMs

Plans for CICS for MVS/ESA programs do not include a VisualAge Generator Server for MVS, VSE, and VM DBRM.

Using MVS/TSO and MVS batch DBRMs

For MVS/TSO and MVS batch programs that use SQL databases, but not DL/I databases, each plan must include the database request module (DBRM) ELADBRM4. This module issues SQL COMMIT WORK and ROLLBACK WORK commands.

For MVS batch, a DBRM is not required for programs that use both DL/I and SQL. If DL/I is used, DL/I function calls are used instead of SQL commands to commit and rollback database changes.

MVS/TSO programs cannot include both DL/I and SQL in the same program or run unit.

Using IMS/VS DBRMs

If a DB2 work database is specified when a program is generated for the IMS/VS environment, the DBRM module ELADBRM3 must be included in each DB2 plan that includes the program. A plan including DBRM module ELADBRM3 must be created even if the program does not use any DB2 program databases.

Additional BIND command keywords

For the BIND command keywords, refer to the DB2 commands document for your version of DB2. Some of the DB2 commands that you might want to add to the BIND commands you use are as follows:

OWNER(authorization-id)

Designates the authorization ID of the plan owner

EXPLAIN(YES)

Obtains information about how SQL statements in the plan are run and inserts information into the x.PLAN_TABLE, where x is the authorization plan owner

This information is useful to a database administrator in understanding how a program gains access to tables, and might help solve logic or performance problems.

Sample BIND commands

The examples of BIND commands show how to define the BIND commands so that the program plan is bound whenever any program included in the plan is changed.

Note: The BIND command does not complete successfully until all programs included in the program plan have been generated.

Binding when the first program uses SQL

Consider the situation in which program A is an IMS main transaction program that does the following:

1. Calls program B, which in turn calls program C
2. Using a DXFR statement, transfers to program D, which transfers to program E
3. Transfers to program X using an XFER statement

Assuming that all the programs use DB2 databases in DB2 subsystem DSN and that the main transactions use the DB2 work database, the database

request modules (DBRMs) for programs A, B, C, D, and E must be bound together in a single plan. Program X has a plan of its own, because an XFER statement starts a new transaction.

Because program X is the only SQL program in its transaction, the default BIND command generated from template EFK2MBDB can be used. Figure 75 shows the bind command generated for program X.

```
DSN SYSTEM(DSN)
* BIND IMSVS APPLICATION WITH DB2 ACCESS AND USES DB2 WORK DATABASE
BIND PLAN(X) -
MEMBER(X,ELADBRM3) -
ACT(REP) -
RETAIN -
VALIDATE(BIND) -
ISOLATION(CS)
* OWNER(OWNERGRP)
```

Figure 75. Sample BIND command generated for Program X

Figure 76 shows a sample BIND command for plan A. This command needs to be included in the bind control parts for A, B, C, D, and E:

```
DSN SYSTEM(DSN)
BIND PLAN(A) -
MEMBER(A,B,C,D,E,ELADBRM3) -
ACT(REP) -
RETAIN -
VALIDATE(BIND) -
ISOLATION(CS)
```

Figure 76. Sample BIND command generated for Plan A

Also consider the situation in which program B is called by program F, which also is an SQL program. Figure 77 shows a sample BIND command for program F, which includes programs B and C.

```
DSN SYSTEM(DSN)
BIND PLAN(F) -
MEMBER(F,B,C,ELADBRM3) -
ACT(REP) -
RETAIN -
VALIDATE(BIND) -
ISOLATION(CS)
```

Figure 77. Sample BIND command generated for Program F

The bind control parts for programs B and C must include the BIND commands for both plan A and plan F. Then if either B or C is generated again, each plan that includes B or C is automatically bound again.

Binding when the first program does not use SQL

Consider the situation in which program Q is an CICS for MVS/ESA main transaction program that does not use SQL. Program Q does the following:

1. Calls program R, which uses SQL
2. Transfers using a DXFR statement to program S, which also uses SQL. Program S does not change the segmented transaction ID from the transaction ID for Q

Because program Q does not use SQL, neither a default BIND command nor a DB2 bind step is created for program Q. However, a program plan must still be bound for transaction Q. Figure 78 shows the BIND command for program Q assuming that the transaction name is also Q and you name your program plan the same as the transaction name for CICS for MVS/ESA.

```
DSN SYSTEM(DSN)
BIND PLAN(Q) -
MEMBER(R,S) -
ACT(REP) -
RETAIN -
VALIDATE(BIND) -
ISOLATION(CS)
```

Figure 78. BIND command for Program Q

You must create a bind control part for both programs R and S so whenever either program is generated the bind occurs for the program plan for transaction Q. You must not create a bind control part for program Q because the plan does not need to be bound when program Q is generated, only when programs R or S are generated.

Binding packages instead of plans

You might want to customize the preparation process to use the DB2 BIND PACKAGE function. When using packages, the preparation BIND step binds each program into its own package in the DB2 database. The system administrator must still bind the packages into plans, but the BIND PLAN command is run one time and does not need to be run again each time a package is replaced.

To use packages, do the following:

- Do not specify any bind control files for programs on the workstation.

- Replace each BIND command template with a template that issues BIND PACKAGE instead of BIND PLAN. Figure 79 shows an example of a BIND PACKAGE template.

```

DSN SYSTEM(DSN)
BIND PACKAGE (%MYCOL.%) -
MEMBER(%EZEMBR%) -
ACTION(REPLACE) -
VALIDATE(BIND) -
ISOLATION(CS)
*   OWNER(OWNERGRP)
*   EXPLAIN(YES)

```

Figure 79. BIND PACKAGE template example

Packages are defined in sets called collections and are bound at different database management system locations. Figure 80 shows how the symbol MYCOL can be specified as the /SYMPARM generation for use in Figure 79.

```

'location-name.collection-id'

or

'collection-id'

```

Figure 80. Setting a value for MYCOL

Where *location-name* defaults to the local database management system. %EZEMBR% is the program DBRM name, which is the same name as the program. Figure 81 shows the fully qualified name of the resulting package.

```

location-name.collection-id.package-id

```

Figure 81. The resulting package

Where *package-id* is the DBRM name.

- Define and run your own BIND PLAN commands on the MVS system for each plan required for your set of programs. See “Sample BIND commands” on page 260 for an example showing how to create the BIND PLAN commands. Use the PKLIST keyword instead of the MEMBER keyword to identify the programs to be bound into the plan. Continue to use the MEMBER keyword to include the host services database request

modules (DBRMs) ELADBRM2, ELADBRM3, or ELADBRM4 in plans that require them. The PLAN needs to be bound one time, no matter how many times the package is replaced.

Use the qualified package name format when identifying packages in the PKLIST. The *location-name* value is optional and defaults to the local database management system. The *collection-id* value can be an asterisk (*) to indicate the collection is determined at run time. The *package-id* value can be an asterisk (*) to indicate all packages in the collection are to be bound into the plan.

Refer to the DB2 documentation for more information on PKLIST formats and on the BIND commands for packages and plans.

Error return codes on BIND commands

A bind command will return an error code of 8 if at least one of the associated programs has not yet been generated and prepared. Do not use a generated program until all programs that it runs with are successfully prepared, and all plans have successfully been bound and receive a completion code of 0.

Binding OS/2 program plans

If SQL is present in your program, the program is compiled with the SQLBIND option. The SQLBIND compiler directive causes a bind file named *prgmname.BND* to be created.

The .BND file must be bound into an object in the DB2/2 database called the program access plan. You can do this in any of the following ways. No special bind command files are required.

- To create the program access plan when the program is compiled, set the symbolic parameter BINDPARM value equal to 'Y' using the /SYMPARM generation option.
- If you are binding multiple bind files together, use the SQLBIND command.
- If the plan is not valid or does not exist, you can have VisualAge Generator Server bind the program for you in the CICS for OS/2 environment. Refer to the *Server Guide for Workstation Platforms* for more information about running in the CICS for OS/2 environment.

Binding for VSE, OS/400, and VM programs

The steps needed to prepare SQL programs to run on VSE, OS/400, and VM systems are automatically included in the preparation jobs generated for the target system. No special bind command files are required.

Chapter 28. Resource associations part

When you generate a COBOL or C++ (targeted for the CICS for AIX or CICS for Windows NT environments) program that uses serial, relative, or indexed files, or printer maps, you can use a resource association part to associate target-system dependent file characteristics, including the file type and the system file or resource name, with the file name specified in the VisualAge Generator record definition. You can also associate the system print files with the file name EZEPRINT.

The new values are used to associate the file name specified in the record definition, or file name EZEPRINT for the print file, with a physical file or data set in the target environment.

Notes:

1. Resource association parts created for generation are not compatible with resource association files used for the VisualAge Generator Developer test facility.
2. Resource association parts are not used during generation for the C++ native environments (OS/2, AIX, HP-UX, and Windows NT). They are used only during execution.

Refer to the *VisualAge Generator Server Guide for Workstation Platforms* for more information.

Creating resource associations parts

You can create and edit resource association parts using the Resource Associations Editor. You can enter options and values in uppercase or lowercase. All of the options can span lines.

Comments are permitted anywhere in the file. Begin comments with the characters `/*` and end them with the characters `*/`.

The file associations specified in the resource associations part are used by the generation process. During generation, the entries in the resource associations part are checked for validity for the specified file in the target environment. If you do not specify resource association information for a file, default values are used based on the target system and file organization. See "File types supported by environment and record organization" on page 274 for information about the default values.

Using multiple resource associations for a file

If the resource associations part for a file contains multiple entries, the resource association used is the first association where the target system specified for the /SYSTEM generation option matches the value specified for the /SYSTEM option. The values specified for the system options can be exact or wildcard matches.

If no entry for the /SYSTEM option value matches the target system, the first entry that matches the file name but does not have the /SYSTEM option specified is used. If you use the same resource associations part for multiple target systems, but you want different resource associations for different systems, be sure to specify the /SYSTEM option.

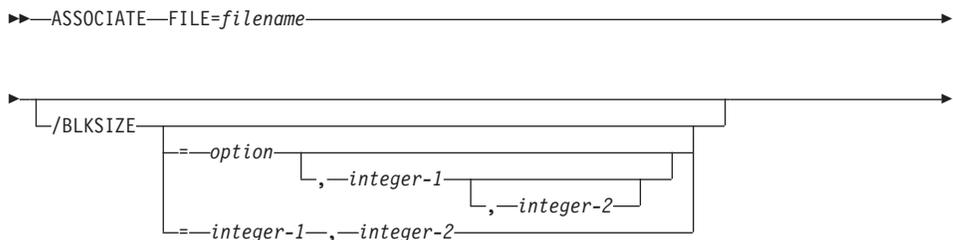
Use the /RESOURCE generation option to specify the name of your resource association part during generation.

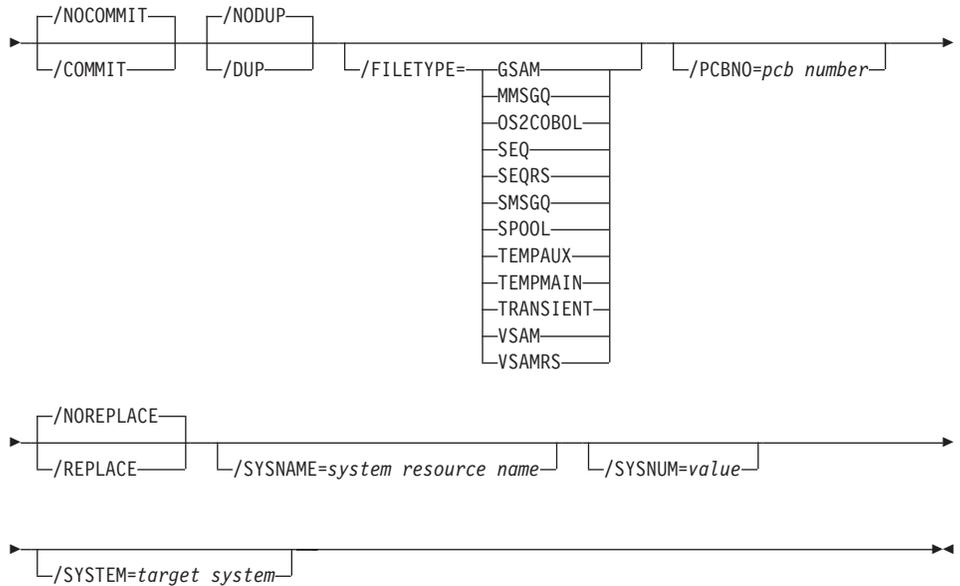
See “/RESOURCE (Resource associations)” on page 338 for more information on specifying a resource association part.

Note: If your program does file or printer I/O, a resource associations part is required. If your program requires a resource associations part, but you want to use the system defaults, you can specify the name of a file that does not contain any resource associations for the /RESOURCE generation option.

Resource association part syntax

The following diagram shows the syntax of the entries in the resource associations part.





ASSOCIATE

Associates the file name specified in a record definition or the printer file with the physical file that is used for the record at run time

FILE Specifies the name of the file for which you are defining system resource association information

The file name can be EZEPRINT, the print file for the program, or one of the file names specified for a serial, indexed, or relative record used by the program.

Wildcard names are permitted. The wildcard character (*) can be specified to be the trailing character in a file name. This enables the project administrator to set up a file naming convention for specifying file types.

/BLKSIZE

Specifies the block size for tape processing on VSE batch for the SEQ file type.

When *integer-1* and *integer-2* are both specified, they represent the minimum and maximum character sizes of the physical record. If only *integer-1* is specified, it specifies the exact character size of the physical record. The default is the record size.

Valid values for *option* are CHAR and REC. If you omit the `/BLKSIZE` option, the default is CHAR. If you specify `/BLKSIZE` without *option*,

the default is REC. *integer-1* and *integer-2* are unsigned nonzero integers. *integer-2* must be equal to or greater than 1. *integer-1* must be less than *integer-2*.

/NOCOMMIT

/NOCOMMIT disables commitment control file management.
/COMMIT enables commitment control file management.

When commitment control is enabled, use the special function words EZECOMIT and EZEROLLB in the program to control commit and rollback boundaries.

This option applies to the OS/400 environment.

The default is /NOCOMMIT.

Refer to the following for more information about commitment control services:

- *ILE COBOL/400 Programmer's Guide 3.1*
- *VisualAge Generator Design Guide*
- *VisualAge Generator Client/Server Communications Guide*

/NODUP

/NODUP specifies that a file cannot contain duplicate record keys.
/DUP specifies that a file can contain duplicate record keys. You must not specify /DUP with the DDS file level keyword UNIQUE in the DDS describing the associate file, which is the target physical file in an OS/400 environment.

Use only the following combinations of the DDS contents and associate option:

UNIQUE /NODUP

no UNIQUE keyword
 /DUP

Table 68 shows the errors you receive if the specifications are reversed.

Table 68. Errors from using the wrong generation options with the UNIQUE keyword

File level keyword	Generation option	COBOL return code after OPEN return code	VisualAge Generator return code after OPEN return code	VisualAge Generator mnemonic
UNIQUE	/DUP	95	220	FMT
no UNIQUE keyword	/NODUP	95	220	FMT

Any other combination will cause an error when the file is opened.

This option applies to the OS/400 environment.

The default is /NODUP.

Refer to the following for more information about commitment control services:

- *ILE COBOL/400 Programmer's Guide 3.1*
- *VisualAge Generator Design Guide*
- *VisualAge Generator Client/Server Communications Guide*

/FILETYPE

Specifies the implementation of the file in a target environment

The valid file types are system-dependent. During the generation process, the file type you specified is checked for validity for the target run-time environment.

Table 69 on page 275 shows the valid file types for each environment. An asterisk (*) indicates that the file type is the default for that record organization in that environment.

The file type can be one of the following:

GSAM

Specifies a serial or print file associated with a GSAM database on IMS BMP or MVS batch

MMSGQ

Specifies a serial or print file associated with a multisegment message queue on IMS/VS or IMS BMP

OS2COBOL

Specifies an indexed, relative, serial, or print file associated with a native COBOL data file on OS/2

COBOL READ/WRITE statements are generated to gain access to the file.

SEQ

Specifies a serial or print file associated with a system sequential file for the CICS for AIX, CICS for Windows NT, IMS BMP, MVS/TSO, MVS batch, OS/400, VM CMS, and VM batch environments

For VSE batch, specifies one of the following:

- An output file directed to a printer (VSE/POWER LST queue member)
- A sequential file in non-VSAM managed space
- A sequential file in VSAM managed space
- A sequential file on tape

For COBOL programs, COBOL READ/WRITE statements are generated to gain access to the file.

SEQRS

Specifies a serial or print file associated with a system sequential file

Calls to host services are generated for all reads and writes to the file. For MVS/TSO, MVS batch, VM CMS, and VM batch target environments, specify SEQRS when using dynamic allocation, the special function word EZEDEST, or the special function word EZEDESTP. This file type is not valid for the OS/400 environment.

SMSGQ

Specifies a serial or print file associated with a single-segment message queue on IMS/VS or IMS BMP

SPOOL

For CICS for MVS/ESA, specifies a serial or print file associated with the JES SPOOL file

For VSE batch, specifies a serial output or print file associated with a VSE/POWER member

For CICS for VSE/ESA, specifies a serial output, serial input, or print file associated with a VSE/POWER queue member

This option can be used with the EZEDEST and EZEDESTP special function words except for the OS/400 environment. For OS/400 environments, SPOOL can only be used with the EZEDESTP special function word.

TEMPAUX

Specifies a relative or serial file associated with an auxiliary temporary storage queue in CICS environments

TEMPMAIN

Specifies a relative or serial file associated with a main temporary storage queue in CICS environments

TRANSIENT

Specifies a serial or print file associated with a transient data queue in CICS environments

VSAM

Specifies a serial, indexed, or relative file associated with a VSAM file on a host system, or defined with a CICS file control table entry (FCT) on a CICS for OS/2 or a file definition (FD) on a CICS for AIX or CICS for Windows NT system.

COBOL READ/WRITE statements or CICS commands are generated to gain access to the file.

VSAMRS

Specifies an indexed, relative, or serial file associated with a VSAM file in non-CICS environments

Calls to host services are generated for all reads and writes to the file. Specify VSAMRS when using the SCANBACK process option, the special function word EZEDEST, or when using dynamic allocation in the MVS/TSO, MVS batch, VM CMS, or VM batch environments. This file type is not valid for the OS/400 environment.

/NOREPLACE

/NOREPLACE specifies that an existing serial file is appended to.

/REPLACE specifies that an existing serial file is replaced.

/REPLACE can be specified when the file type is SEQ and the target environment is one of the following:

- AIX
- AIXCICS
- CICSNT
- HP
- OS2

Note: The resource association file is only referenced at run time when the target environment is AIX, HP-UX, OS2, or Windows NT.

When you specify /REPLACE, the first ADD to a serial file in the program or the first ADD to a serial file following a CLOSE in the program adds data to the beginning of the serial file; replacing all previous contents.

The default is /NOREPLACE.

/PCBNO

Specifies the number of the program control block (PCB) in the program specification block (PSB) that represents the message queue or GSAM database associated with the file

The number is an integer that ranges from 0 to the number of PCBs in the PSB.

Use /PCBNO only if the file type is SMSGQ, MMSGQ, or GSAM. This option is ignored with other file types.

If you do not specify the /PCBNO option, the following defaults are set at generation time by the PSB definition and the use of the file in the program:

- 0 for SMSGQ and MMSGQ input files
- 1 for SMSGQ and MMSGQ output files
- The first GSAM PCB for GSAM files

This option is not valid for the OS/400 environment.

/SYSNAME

Specifies the system name of the file or data set associated with the file

The format of the name is system- and file-type-dependent. Table 69 on page 275 shows the valid file types for each environment. An asterisk (*) indicates that the file type is the default for that record organization in that environment.

Symbolic parameters are permitted in the /SYSNAME option specification. See “Chapter 12. Symbolic parameters” on page 123 for more information about symbolic parameters. If the symbolic parameter EZEDD is used in the /SYSNAME option specification, the file name specified for the FILE option on the association is substituted in place of EZEDD. If any special characters are included in the value, enclose the value in single quotation marks (') or double quotation marks ("). Figure 82 shows the special characters that require quotation marks.

% = , () / or a space character

Figure 82. Characters that require quotation marks

/SYSNUM

Specifies the system logical unit number assigned the device on which the sequential file resides

The valid range of values is from 000–254; the default is 005.

This option is only valid for SEQ files on VSE batch.

Note: If multiple VSE batch SEQ files need to have default system logical unit numbers set, default values are generated in sequential order starting with 005.

Default system logical unit numbers are set at generation time. It is possible that multiple programs in a run unit might be generated with some of the same default system logical unit numbers. If you want to

check the default system logical unit numbers assigned, generate a source listing using the /LISTING generation option.

A default system logical unit number is generated for EZEPRINT even when the intended use of EZEPRINT is as filetype SPOOL. The map group for EZEPRINT might be shared by multiple programs in a run unit. The /SYSNUM value within the map group is equivalent to the /SYSNUM value specified or defaulted to at the time the last program in the run unit was generated.

/SYSTEM

Specifies the target system affected by this associate command

You might have multiple associates in one resource association file. The resource association used is the first association where the value specified for the /SYSTEM option matches the value specified for the /SYSTEM generation option value. The values specified for the system options can be exact or wildcard matches. If there is not an entry where the /SYSTEM value matches the target system specified for the GENERATE subcommand, then the first entry in the resource association file that matches the file name but does not have the /SYSTEM option specified is used.

The target system value can be one of the following:

- AIXCICS
- IMSBMP
- IMSVS
- MVSBATCH
- MVSCICS
- NTCICS
- OS2CICS
- OS400
- SOLACICS
- TSO
- VMCMS
- VMBATCH
- VSEBATCH
- VSECICS
- WINNT

The following target systems use the resource association file at run time. Refer to the *VisualAge Generator Server Guide for Workstation Platforms* for information on these target systems.

- OS/2
- AIX
- HP-UX
- Windows NT

Sample resource associations part

Figure 83 shows a sample resource associations part.

```
/* *****  
/* This is a sample resource association part that uses naming conventions on the file */  
/* name to set the file type to a value other than the usual default. In this example, */  
/* the target systems being used are IMSBMP, IMSVS, MVSCICS, OS2CICS, MVS BATCH, VMCMS, */  
/* and OS/400. */  
/* */  
/* The defaults are changed in the following ways: */  
/* */  
/* For MVSCICS, files with names starting with a 'T' default to type TEMPMAIN */  
/* */  
/* For all targets, files with names starting with a 'V' default to type VSAM */  
/* */  
/* For MVS BATCH, files with names starting with a 'G' default to type GSAM */  
/* */  
/* For OS2CICS, the default file type is always OS2COBOL (unless the name starts */  
/* with a 'V') */  
/* */  
/* In this example, TESTHLQ is a symbolic parameter that is set when generating for */  
/* MVS BATCH. TESTHLQ includes the high-level qualifier used for test data sets. */  
/* *****  
/* any files that need values other than the defaults given below need to have their */  
/* own associations above this line */  
  
ASSOCIATE FILE=T* /SYSTEM=MVSCICS  
/FILETYPE=TEMPMAIN /SYSNAME='%EZEDD%'  
ASSOCIATE FILE=V*  
/FILETYPE=VSAM /SYSNAME='%EZEDD%'  
ASSOCIATE FILE=V* /SYSTEM=MVS BATCH  
/FILETYPE=VSAM /SYSNAME='%TESTHLQ%.VSAM.%EZEDD%'  
ASSOCIATE FILE=G* /SYSTEM=MVS BATCH  
/FILETYPE=GSAM /SYSNAME='%TESTHLQ%.GSAM.%EZEDD%'  
ASSOCIATE FILE=* /SYSTEM=OS2CICS  
/FILETYPE=OS2COBOL /SYSNAME='C:\TESTDATA\%EZEDD%.DAT'  
ASSOCIATE FILE=FILE1 /SYSTEM=IMSVS  
/FILETYPE=MMSGQ  
ASSOCIATE FILE=FILE1 /SYSTEM=MVS BATCH  
/FILETYPE=SEQR /SYSNAME='THE.NAME.OF.MY.FILE'  
ASSOCIATE FILE=FILE2 /SYSTEM=MVS BATCH  
/FILETYPE=GSAM /PCBNO=3  
ASSOCIATE FILE=FILE2 /SYSTEM=IMSBMP  
/SYSNAME='THE.NAME.OF.MY.OTHER.FILE'  
ASSOCIATE FILE=FILE3 /SYSTEM=OS2CICS  
/FILETYPE=OS2COBOL  
ASSOCIATE FILE=COB* /SYSTEM=OS2CICS  
/FILETYPE=OS2COBOL  
ASSOCIATE FILE=EZEPRINT /SYSTEM=OS400  
/FILETYPE=SPOOL  
ASSOCIATE FILE=FILE4 /SYSTEM=VMCMS  
/FILETYPE=SEQR /SYSNAME='FILE4NAM FILE4TYP A1'
```

Figure 83. Sample resource associations part

File types supported by environment and record organization

Table 69 on page 275 shows the valid file types for each environment.

Table 69. File types supported by environment and record organization

System	Indexed	Relative	Serial	Print
CICS for AIX	VSAM *	VSAM * TEMPMAIN TEMPAUX	SEQ * VSAM TRANSIENT TEMPMAIN TEMPAUX	SEQ *
CICS for Windows NT	VSAM *	VSAM * TEMPMAIN TEMPAUX	SEQ * VSAM TRANSIENT TEMPMAIN TEMPAUX	SEQ *
CICS for OS/2	VSAM * OS2COBOL	VSAM * TEMPMAIN TEMPAUX OS2COBOL	OS2COBOL * VSAM TRANSIENT TEMPMAIN TEMPAUX	OS2COBOL *
IMS/VS	Not available	Not available	SMSGQ * MMSGQ	SMSGQ *
IMS BMP	VSAMRS * VSAM	VSAMRS * VSAM	SEQRS * VSAMRS SEQ VSAM SMSGQ MMSGQ GSAM	SEQRS * SEQ SMSGQ GSAM
MVS batch	VSAMRS * VSAM	VSAMRS * VSAM	SEQRS * VSAMRS SEQ VSAM GSAM	SEQRS * SEQ GSAM
CICS for MVS/ESA	VSAM *	VSAM * TEMPMAIN TEMPAUX	VSAM * TRANSIENT TEMPMAIN TEMPAUX SPOOL	TRANSIENT * SPOOL
MVS/TSO	VSAMRS * VSAM	VSAMRS * VSAM	SEQRS * VSAMRS SEQ VSAM	SEQRS * SEQ
OS/400	VSAM	VSAM	SEQ * VSAM	SEQ * SPOOL
VM CMS	VSAMRS * VSAM	VSAMRS * VSAM	SEQRS * VSAMRS SEQ	SEQRS * SEQ

Table 69. File types supported by environment and record organization (continued)

System	Indexed	Relative	Serial	Print
VM batch	VSAMRS * VSAM	VSAMRS * VSAM	SEQRS * VSAMRS SEQ	SEQRS * SEQ
VSE batch ¹	VSAMRS * VSAM	VSAMRS * VSAM	SPOOL ² SEQ ³ VSAMRS VSAM	SPOOL * SEQ
CICS for VSE/ESA	VSAM *	VSAM * TEMPMAIN TEMPAUX	VSAM * TRANSIENT TEMPMAIN TEMPAUX SPOOL	TRANSIENT * SPOOL

Notes:

An asterisk (*) indicates that the file type is the default for that record organization in that environment.

¹SPOOL is not a valid file type for VSE batch serial input files.

²SPOOL is the default if the file is an output file.

³SEQ is the default if the file is an input file.

File types supported by CICS environments

Table 70 shows a list of the file types and their implementation supported in the CICS environments, including CICS for OS/2, CICS for AIX, CICS for Windows NT, CICS for MVS/ESA, and CICS for VSE/ESA.

Table 70. File type keywords supported by CICS environments

Environment	File implementation	System resource name format and length	Default system resource name
OS2COBOL			

Table 70. File type keywords supported by CICS environments (continued)

Environment	File implementation	System resource name format and length	Default system resource name
CICS for OS/2	<p>An indexed, relative, serial, or print file associated with a COBOL data file on OS/2. COBOL READ/WRITE statements are generated to gain access to the file.</p> <p>File sharing for COBOL-managed data files is not supported in programs generated for CICS for OS/2. Whenever the file is opened, an exclusive lock is obtained on the file until it is closed.</p>	OS/2 file name, 65 bytes	For print files, the default system resource name is LPT1; for all other file types, the VisualAge Generator file name.
SEQ			
CICS for AIX	An AIX or print file	<p>The file name must be one of the following:</p> <ul style="list-style-type: none"> • A fully-qualified AIX file name • The name of a file located in a directory specified by the FCWDPATH environment variable of the program process <p>The maximum length is 255 bytes.</p>	For print files, the default system resource name is EZEPRINT; for all other file types, the VisualAge Generator file name.
CICS for Solaris	A Solaris or print file	<p>The file name must be one of the following:</p> <ul style="list-style-type: none"> • A fully-qualified Solaris file name • The name of a file located in a directory specified by the FCWDPATH environment variable of the program process <p>The maximum length is 255 bytes.</p>	For print files, the default system resource name is EZEPRINT; for all other file types, the VisualAge Generator file name.

Table 70. File type keywords supported by CICS environments (continued)

Environment	File implementation	System resource name format and length	Default system resource name
CICS for Windows NT	A CICS for Windows NT or print file	<p>The file name must be one of the following:</p> <ul style="list-style-type: none"> • A fully-qualified CICS for Windows NT file name • The name of a file located in a directory specified by the FCWDPATH environment variable of the program process <p>The maximum length is 255 bytes.</p>	For print files, the default system resource name is EZEPRINT; for all other file types, the VisualAge Generator file name.
CICS for MVS/ESA	A serial or print file associated with a JES SPOOL file	<p>OS/2 file name, 65 bytes</p> <p>Input file: The name is in the format <i>userid.class</i>, where the <i>userid</i> is a 4- to 8-character external writer name, and <i>class</i> is a 1-character spool class. CICS requires that the first 4 characters of the external writer name be the same as the first 4 characters of the CICS APPLID value used to identify the CICS region to ACF/VTAM. The <i>class</i> value is optional and its default is "A". The maximum name size is 10 bytes.</p> <p>Output file: The name is in the format <i>nodeid.userid.class</i>, where <i>nodeid</i> is a 1- to 8-character system node ID, <i>userid</i> is a 1- to 8-character system user ID, and <i>class</i> is a 1-character spool class. The <i>class</i> value is optional and its default is "A". If <i>class</i> is not specified, <i>userid</i> is also optional and its default is the CICS user ID. This is the same value as stored in the EZEUSRID special function word. The maximum name size is 19 bytes.</p> <p>Refer to the CICS customization guide for more information on specifying the system resource name.</p>	For print files, the default system resource name is EZEP; for serial files, the VisualAge Generator file name.

Table 70. File type keywords supported by CICS environments (continued)

Environment	File implementation	System resource name format and length	Default system resource name
CICS for VSE/ESA	A serial output, serial input, or print file associated with a VSE/POWER queue member	<p>Input file: The name is in the format <i>userid.class</i>, where the <i>userid</i> is a 4- to 8-character external writer name, and <i>class</i> is a 1-character spool class. See “Output file naming format for CICS for VSE/ESA” on page 283.</p> <p>CICS requires that the first 4 characters of the external writer name be the same as the first 4 characters of the CICS APPLID value used to identify the CICS region to ACF/VTAM. An asterisk (*) can be used as the <i>userid</i> value, in which case the default is the contents of the EZEUSRID special function word. The <i>class</i> value is optional and, if left blank, its default is “A”. You cannot use an asterisk to request the default class for input spool file. The maximum name size is 10 bytes.</p> <p>The input spool file is read from the PUN VSE/POWER queue.</p> <p>Output file: The name is in the format: <i>jobname.queue.class.disp.form.node.userid.parm.</i> (This name must be typed without spaces) or for VSE/POWER PRT, the name is in the format: <i>jobname.queue.class.disp.form.node.userid.fcbname.copies.</i> See “Output file naming format for CICS for VSE/ESA” on page 283.</p>	<p>The default system resource name is the VisualAge Generator file name used as the SPOOL name. The value for <i>jobname</i> is the VisualAge Generator file name, and all other values have the specified defaults.</p>
TEMPAUX			

Table 70. File type keywords supported by CICS environments (continued)

Environment	File implementation	System resource name format and length	Default system resource name
All CICS	<p>A relative or serial file associated with an auxiliary temporary storage queue in CICS environments.</p> <p>TEMPAUX files can be used by only one transaction at a time. The program enqueues, by issuing the ENQ command with the NOSUSPEND option, when the queue is first opened on resource name EZETEMP-queueName. The program dequeues by issuing the DEQ command, when the file is closed or when recoverable resources are committed. A file is closed because of a CLOSE process option or when the end of the program is reached.</p> <p>A control byte is added to each temporary storage record written by generated programs. Refer to the <i>VisualAge Generator Design Guide</i> for more information about using temporary storage.</p>	Queue name, 8 bytes	The default system resource name is the VisualAge Generator file name.
TEMPMAIN			

Table 70. File type keywords supported by CICS environments (continued)

Environment	File implementation	System resource name format and length	Default system resource name
All CICS	<p>A relative or serial file associated with a main temporary storage queue in CICS environments.</p> <p>TEMPMAIN files can be used by only one transaction at a time. The program enqueues, by issuing the ENQ command with the NOSUSPEND option, when the queue is first opened on resource name EZETEMP-queuename. The program dequeues, by issuing the DEQ command, when the file is closed or when recoverable resources are committed. A file is closed because of a CLOSE process option or when the end of the program is reached.</p>	<p>Queue name, 8 bytes</p> <p>In the current releases of CICS for OS/2, files defined as main storage queues to CICS are actually implemented as auxiliary storage queues.</p> <p>A control byte is added to each temporary storage record written by generated programs. Refer to the <i>VisualAge Generator Design Guide</i> for more information about using temporary storage.</p>	<p>The default system resource name is the VisualAge Generator file name.</p>
TRANSIENT			

Table 70. File type keywords supported by CICS environments (continued)

Environment	File implementation	System resource name format and length	Default system resource name
All CICS	A serial or print file associated with a transient data queue in CICS environments.	Destination control table (DCT) name or Transient data definition (TDD) name, 4 bytes	<p>For print files, the default system resource name is EZEP; for serial files, the first 4 characters of the VisualAge Generator file name.</p> <p>The EZEDEST and EZEDESTP special function words are initialized with the system resource name associated at generation time.</p> <p>The exception is when the /PRINTDEST generation option is specified with a value of TERMID; then the EZEDESTP special function word is initialized to the value of EIBTRMID.</p>
VSAM			
CICS for AIX and CICS for Solaris	An indexed, relative, or serial file associated with a CICS for AIX managed file. EXEC CICS statements are generated for each read or write to the file.	File definition (FD) name, 8 bytes	The default system resource name is the VisualAge Generator file name.
VSAM			

Table 70. File type keywords supported by CICS environments (continued)

Environment	File implementation	System resource name format and length	Default system resource name
CICS for Windows NT	An indexed, relative, or serial file associated with a CICS for Windows NT managed file. EXEC CICS statements are generated for each read or write to the file.	File definition (FD) name, 8 bytes	The default system resource name is the VisualAge Generator file name.
CICS for OS/2	An indexed, relative, or serial file associated with a CICS for OS/2 managed file. EXEC CICS statements are generated for each read or write to the file.	File control table (FCT) name, 8 bytes	The default system resource name is the VisualAge Generator file name.
CICS for MVS/ESA and CICS for VSE/ESA	An indexed, relative, or serial file associated with a VSAM file. EXEC CICS statements are generated for each read or write to the file.	File control table (FCT) name, 8 bytes on MVS and 7 bytes on VSE	For CICS for MVS/ESA, the default system resource name is the VisualAge Generator file name; for CICS for VSE/ESA, the first 7 characters of the VisualAge Generator file name.

Output file naming format for CICS for VSE/ESA

The *jobname* parameter must be specified, or the default must be explicitly requested by specifying an asterisk (*). All other parameters can specify that the default value can be used by specifying an asterisk (*) or a blank. However, if a default value for a parameter is specified using a blank, the default values are used for all subsequent parameters. The components of the output file name are defined as follows:

jobname The 1- to 8-character name that defines the jobname for the VSE/POWER queue member. This value is used in all cases except when the CICS Report Control Facility (RCF) is not being used, for example when the queue is PUN or LST. For these two cases, the value in *jobname* is ignored, and the VSE/POWER queue member jobname is the CICS for VSE/ESA

program ID. For all other cases, an asterisk (*) in this field causes the VisualAge Generator file name to be used for the record.

queue The 3 characters that identify the destination VSE/POWER queue for the file. The destinations, by output type, are as follows:

RDR for job output
LST for list output
PUN for punch output
PRT for list output (using RCF)

Using any other characters for *queue* makes the spool name not valid. An asterisk (*) or a blank in this field causes the PRT queue to be used. The types of support are RCF and basic CICS.

The RCF is used for files that specify RDR or PRT in this field.

Basic CICS SPOOL support is used for files that specify PUN or LST in this field.

If you attempt to use RCF when you do not have RCF installed on your CICS system, CICS returns an error message. This might be an AEY9 transaction abend, a NOSPOOL condition, or the message "SPOOLING SYSTEM IS NOT AVAILABLE".

When the *queue* value is PRT or LST, the file is opened with the ASA option. This option specifies that the report is created using an American National Standard printer-control character at the beginning of each line of data. If you are using a serial file, you must ensure that valid carriage control characters are used. If the file is a print file, the American National Standard printer-control characters are automatically added for you by VisualAge Generator Server for MVS, VSE, and VM.

class The single character that specifies class. An asterisk (*) or blank in this field causes the default of 'A' to be used.

disp The single character that specifies the VSE/POWER disposition status of the queue member after it is closed. Valid values for *disp* are as follows:

D Process the job and delete it after processing
H Hold the job in the queue until released
K Process the job and keep it in the queue after processing
L Let the job stay in the queue until released

Using any other characters for *disp* makes the spool name not valid. This field is not applicable when the RCF is not in use, for example when the *queue* is LST or PUN. An asterisk (*) or a blank in this field causes the default of **D** to be used.

form The 4 characters that identify the form number for print output. An asterisk (*) or a blank in this field causes the default of your location's standard form to be used. This field is applicable when the *queue* is PRT and it is ignored for all other queues.

node The 1- to 8-characters that specify the system node ID. An asterisk (*) or a blank in this field causes the default of the current system node ID to be used.

userid The 1- to 8-characters that specify the user ID. An asterisk (*) or a blank in this field and you are signed on to CICS, the default value of *userid* is the contents of the special function word EZEPID.

An asterisk (*) or a blank in this field and you are not signed on to CICS, the default value of *userid* is an asterisk (*).

parm The string of characters used to specify output operands for files on the VSE/POWER LST queue. This option is used only if the *queue* is LST, and it is ignored for all other queues. This string is passed to CICS in the OUTDESCR option of the CICS for VSE/ESA SPOOLOPEN OUTPUT command. The characters must be specified in the correct format for the OUTDESCR option. The parameters use the same keywords and values as are used on the VSE/POWER LST statement for program-user-defined output operands but the syntax varies slightly. Figure 84 shows how you can use FORMDEF FORM1 and PAGEDEF PAGE1, the *parm* string.

```
'FORMDEF(FORM1) PAGEDEF(PAGE1)'
```

Figure 84. FORMDEF example

Figure 85 shows how the spool file might appear.

```
JOBNAME1.LST.*.*.*.*.*.FORMDEF(FORM1) PAGEDEF(PAGE1).
```

Figure 85. Spool file

The calculating and inserting of the length area required by CICS for VSE/ESA at the beginning of the string is handled automatically. The length of the *parm* string is variable and depends on the length of the spool file specification up to this point. The total length of the spool file specification cannot be over 65 characters.

fcbname The name of the FCB-image phase which VSE/POWER to use for printing the related job output. The name phase must be cataloged in a sublibrary accessible from the VSE/POWER partition. The name can be up to eight alphanumeric (characters, numbers, and special characters). If omitted, the system default FCB is used. The default name can be specified with an asterisk.

copies The number of copies to be printed from the print queue. The value can be from 1 to 255. If omitted, VSE/POWER prints one copy. The default value is 1.

File types supported for MVS/TSO

Table 71 shows a list of file types supported for the MVS/TSO environment.

Table 71. File type keywords supported by the MVS/TSO environments

File implementation	System resource name format and length	Default system resource name
SEQ		
A serial or print file associated with a system sequential file. COBOL READ/WRITE statements are generated to gain access to the file. SEQ must not be specified when using dynamic allocation, the special function word EZEDEST, or the special function word EZEDESTP.	Fully qualified data set name, 54 bytes	For print files, the default system resource name is EZEPRINT; for serial files, the VisualAge Generator file name.
The file name is used as the DD name. For serial files, the system resource name is used as the data set name in the generated CLIST. For print files, the system resource name is not used in the generated CLIST.		
SEQRS		
A serial or print file associated with a system sequential file. Calls to host services are generated for all reads and writes to the file. SEQRS must be specified when using dynamic allocation, the special function word EZEDEST, or the special function word EZEDESTP.	Fully qualified data set name, 54 bytes EZEDEST/EZEDESTP: fully-qualified data set name, 54 bytes, or DD name, 8 bytes	For print files, the default system resource name is EZEPRINT; for serial files, the VisualAge Generator file name.
The system resource name is used as the data set name in the generated CLIST. Refer to the <i>VisualAge Generator Design Guide</i> for more information about dynamically allocating files.		The EZEDEST and EZEDESTP special function words are initialized with the system resource name associated at generation time.
VSAM		

Table 71. File type keywords supported by the MVS/TSO environments (continued)

File implementation	System resource name format and length	Default system resource name
<p>An indexed, relative, or serial file associated with a VSAM file. COBOL READ/WRITE statements are generated to gain access to the file. VSAM must not be specified when using dynamic allocation, the SCANBACK process option, or the EZEDEST special function word.</p> <p>The file name is used as the DD name. The system resource name is used as the data set name in the generated CLIST.</p>	Fully qualified data set name, 44 bytes	The default system resource name is the VisualAge Generator file name.
VSAMRS		
<p>An indexed, relative, or serial file associated with a VSAM file. Calls to host services are generated for all reads and writes to the file.</p> <p>The system resource name is used as the data set name in the generated CLIST. Refer to the <i>VisualAge Generator Design Guide</i> for more information about dynamically allocating files.</p>	<p>Fully qualified data set name, 44 bytes</p> <p>EZEDEST: fully qualified data set name, 44 bytes, or DD name, 8 bytes</p>	<p>The default system resource name is the VisualAge Generator file name.</p> <p>The EZEDEST special function word is initialized with the system resource name associated at generation time.</p>

File types supported for IMS BMP, IMS/VS, and MVS batch

Table 72 shows a list of file types supported for the MVS batch, IMS/VS, and IMS BMP environments. Listed under each file type are the environments supporting that file type.

Note: For the IMS BMP environment, all programs generated as COBOL programs that run together in the same job step must associate EZEPRINT with either the message queue or a sequential or GSAM file.

Table 72. File type keywords supported by IMS BMP, IMS/VS, and MVS batch environments

Environment	File implementation	System resource name format and length	Default system resource name
GSAM			

Table 72. File type keywords supported by IMS BMP, IMS/VS, and MVS batch environments (continued)

Environment	File implementation	System resource name format and length	Default system resource name
IMS BMP MVS batch	A serial or print file associated with a GSAM database.	Data set name, 44 bytes	For print files, the default system resource name is EZEPRINT; for serial files, the VisualAge Generator file name.
MMSGQ			
IMS BMP IMS/VS	A serial file associated with a multiple segment message queue.	The logical terminal name or transaction code associated with the message queue, 8 bytes	The default system resource name is the VisualAge Generator file name.
SEQ			
IMS BMP MVS batch	A serial or print file associated with a system sequential file. COBOL READ/WRITE statements are generated to gain access to the file. SEQ must not be specified when using dynamic allocation, the special function word EZEDEST, or the special function word EZEDESTP. The file name is used as the DD name. For serial files, the system resource name is used as the data set name in the generated JCL. For print files, the system resource name is not used in the generated JCL.	Data set name, 54 bytes	For print files, the default system resource name is EZEPRINT; for serial files, the VisualAge Generator file name.
SEQRS			
IMS BMP MVS batch	A serial or print file associated with a system sequential file. Calls to host services are generated for all reads and writes to the file. SEQRS must be specified when using dynamic allocation, the special function word EZEDEST, or the special function word EZEDESTP. The system resource name is used as the data set name in the generated JCL. Refer to the <i>VisualAge Generator Design Guide</i> for more information about dynamically allocating files.	Generation: data set name, 54 bytes EZEDEST/EZEDESTP: data set name, 54 bytes, or DD name, 8 bytes	For print files, the default system resource name is EZEPRINT; for serial files, the VisualAge Generator file name. The EZEDEST and EZEDESTP special function words are initialized with the system resource name associated at generation time.

Table 72. File type keywords supported by IMS BMP, IMS/VS, and MVS batch environments (continued)

Environment	File implementation	System resource name format and length	Default system resource name
SMSGQ			
IMS BMP IMS/VS	A serial or print file associated with a single-segment message queue.	The logical terminal name or transaction code associated with the message queue, 8 bytes	For print files, the default system resource name is EZEPRINT; for serial files, the VisualAge Generator file name.
VSAM			
IMS BMP MVS batch	An indexed, relative, or serial file associated with a VSAM file. COBOL READ/WRITE statements are generated to gain access to the file. VSAM must not be specified when using dynamic allocation, the SCANBACK process option, or the special function word EZEDEST. The file name is used as the DDname. The system resource name is used as the data set name in the generated JCL.	Data set name, 44 bytes	The default system resource name is the VisualAge Generator file name.
VSAMRS			
IMS BMP MVS batch	An indexed, relative, or serial file associated with a VSAM file. Calls to host services are generated for all reads and writes to the file. VSAMRS must be specified when using dynamic allocation, the SCANBACK process option, or the special function word EZEDEST. The system resource name is used as the data set name in generated JCL. Refer to the <i>VisualAge Generator Design Guide</i> for more information about dynamically allocating files.	Generation: data set name, 44 bytes EZEDEST: data set name, 44 bytes, or DD name, 8 bytes	The default system resource name is the VisualAge Generator file name. The EZEDEST special function word is initialized with the system resource name associated at generation time.

File types supported by OS/400

Table 73 on page 290 shows a list of file types supported for the OS/400 environment.

Table 73. File type keywords supported by the OS/400 environment

File implementation	System resource name format and length	Default system resource name
SEQ		
For serial files, a file associated with a system sequential file, which is a physical or logical file in "arrival sequence" organization. COBOL READ/WRITE statements are generated to access the file. For printer files, SEQ and SPOOL differ in that SEQ inserts a page eject at the end of the print stream when the file is closed.	Unqualified file name, 10 bytes maximum. For serial files, the run-time library list (*LIBL) and the first member in the physical or logical file is used to complete qualification of the file name. Override Database File commands can be used prior to run time to alter the resolution to the file; use EZEDEST at run time. For printer files, same as for SPOOL	For serial files, the default system resource name is the VisualAge Generator file name. For printer files, same as for SPOOL
SPOOL		
A printer file object (*PRTF) such as QVGNPRTF, which is shipped with VisualAge Generator Server for AS/400	Unqualified file name, 10 bytes maximum. The run-time library list (*LIBL) is used to complete qualification of the file name. Override Printer File commands can be used prior to run time to alter the resolution to the file; use EZEDESTP at run time.	For printer files (EZEPRINT), the default system resource name is QVGNPRT. Output from EZEPRINT is sent to the job's output queue as a spooled file that is named with the system resource name.
VSAM		
OS/400 database files, physical or logical, in indexed or "arrival sequence" organization SEQ (depending on the VisualAge Generator corresponding record organization)	Same as SEQ	Same as SEQ
COBOL READ/WRITE statements are generated to access the file. To use relative record access, you need to preallocate the file.		

File types supported for VM CMS and VM batch

Table 74 on page 291 shows a list of file types supported for the VM environments.

Table 74. File type keywords supported by the VM environments (continued)

File implementation	System resource name format and length	Default system resource name
<p>An indexed, relative, or serial file associated with a VSAM file. COBOL READ/WRITE statements are generated to access the file.</p> <p>The system resource name is used as the DLBL file identifier in the generated run-time exec. File connections are performed using the file name as the DLBL file name. VSAM must not be specified when you are using the SCANBACK process option or the EZEDEST special function word.</p>	<p>DLBL file identifier, 44 bytes</p>	<p>The default system resource name is the first 7 characters of the VisualAge Generator file name.</p> <p>The EZEDEST and EZEDESTP special function words are initialized with the system resource name associated at generation time.</p>
VSAMRS		
<p>An indexed, relative, or serial file associated with a VSAM file. Calls to host services are generated for all reads and writes to the file.</p> <p>The system resource name is used as the DLBL file identifier in the generated run-time exec. VSAMRS must be specified when you use the SCANBACK process option or the EZEDEST special function word.</p>	<p>Generation: DLBL file identifier, 44 bytes</p> <p>EZEDEST: DLBL file name, 7 bytes</p>	<p>The default system resource name is the first 7 characters of the VisualAge Generator file name.</p> <p>The EZEDEST and EZEDESTP special function words are initialized with the system resource name associated at generation time.</p>

File types supported for VSE batch

Table 75 shows a list of file types supported for the VSE batch environment.

Table 75. File type keywords supported by the VSE batch environment

File implementation	System resource name format and length	Default system resource name
SEQ		

Table 75. File type keywords supported by the VSE batch environment (continued)

File implementation	System resource name format and length	Default system resource name
<p>A system logical unit number is used for all SEQ files. This value is specified using the /SYSNUM option for the resource association.</p>	<p>The first 7 characters of the VisualAge Generator file name, used as the DLBL or TLBL name.</p>	
<p>System Logical Unit Number</p>	<p>Default System Logical Unit Number</p>	
<p>The following shows how to specify <i>nnn</i> as the 3-character system label number.</p>	<p>The default is 005.</p>	
<p>SYSnnn</p>	<p>If multiple VSE batch SEQ files need default system logical unit numbers set, default values are assigned in sequential order starting with 005.</p>	
<p>The valid range of values is 000–254.</p>		
<p>The DLBL or TLBL name is always the first 7 characters of the VisualAge Generator file name associated with that VisualAge Generator record.</p>		
<p>SPOOL</p>		
<p>A sequential file in VSAM managed space</p>	<p>The name is in the format: <i>jobname.queue.class.disp.form.node.userid.fcbname.copy</i> (You must type this name without spaces.) See “Output file naming format for VSE batch” on page 294.</p>	<p>The default system resource name is the VisualAge Generator file name used as the SPOOL name. The value of <i>jobname</i> is the VisualAge Generator file name, and all other values have the specified defaults.</p>
<p>Calls to host services are generated for all writes to the file.</p>		<p>The <i>jobname</i> parameter must be specified, or the default must be explicitly requested by specifying an asterisk (*). All other parameters can specify that the default value can be used by specifying an asterisk (*) or a blank. However, if a default value for a parameter is specified using a blank, the default values are used for all subsequent parameters.</p>
<p>VSAM</p>		

Table 75. File type keywords supported by the VSE batch environment (continued)

File implementation	System resource name format and length	Default system resource name
<p>An indexed, relative, or serial file associated with a VSAM file. COBOL READ/WRITE statements are generated to access the file.</p> <p>The system resource name is used as the DLBL file identifier in the generated JCL. File connections are performed using the file name as the DLBL file name. VSAM must not be specified when using the SCANBACK process option or the EZEDEST special function word.</p>	<p>DLBL file identifier, 44 bytes</p>	<p>The default system resource name is the first 7 characters of the VisualAge Generator file name.</p> <p>The EZEDEST and EZEDESTP special function words are initialized with the system resource name associated at generation time.</p>
VSAMRS		
<p>An indexed, relative, or serial file associated with a VSAM file. Calls to host services are generated for all reads and writes to the file.</p> <p>The system resource name is used as the DLBL file identifier in the generated JCL. VSAMRS must be specified when using the SCANBACK process option or the EZEDEST special function word.</p>	<p>Generation: DLBL file identifier, 44 bytes</p> <p>EZEDEST: DLBL file name, 7 bytes</p>	<p>The default system resource name is the first 7 characters of the VisualAge Generator file name.</p> <p>The EZEDEST special function word is initialized with the system resource name associated at generation time.</p>

Output file naming format for VSE batch

The components of the name in the format

jobname.queue.class.disp.form.node.userid.fcbyname.copy are defined as follows:

jobname The 1- to 8-character name that defines the jobname for the VSE/POWER queue member. This option is used in all cases. An asterisk (*) in this field causes the default of the VisualAge Generator file name for the record to be used.

queue The 3 characters that identify the destination VSE/POWER queue for the file. The destinations and their appropriate output type, follow:

RDR for job output
LST for list output
PUN for punch output

Using any other characters for *queue* makes the spool name not valid. An asterisk (*) or a blank in this field causes the LST queue to be used. The queue PRT can be used, but is changed to LST. The PRT value is not valid for VSE batch. When the *queue* is LST, the file is opened with the ASA option. This option specifies that the report is created using an American National Standard printer-control character at the beginning of each line of data. If you use a serial file, you must ensure that valid carriage control characters are used. If the file is a print file, the American National Standard printer-control characters are automatically added for you by VisualAge Generator Server for MVS, VSE, and VM.

class The single character that specifies class. An asterisk (*) or blank in this field causes the default of 'A' to be used.

disp The single character that specifies the VSE/POWER disposition status of the queue member after it is closed. Valid values for *disp* follow:

- D** Process the job and delete it after processing
- H** Hold the job in the queue until released
- K** Process the job and keep it in the queue after processing
- L** Let the job stay in the queue until released

Using any other characters for *disp* makes the spool name not valid. An asterisk (*) or a blank in this field causes the default of 'D' to be used.

form The 4 characters that identify the form number for print output. An asterisk (*) or a blank in this field causes the default of your location's standard form to be used. This field is applicable when the *queue* is LST, and it is ignored for all other queues.

node The 1- to 8-characters that specify the system node ID. An asterisk (*) or a blank in this field causes the default of the current system node ID to be used.

userid The 1- to 8-characters that specify the user ID. An asterisk (*) or a blank in this field causes the default of the userid 'ANY' to be used.

fcfname The name of the FCB-image phase which VSE/POWER to use for printing the related job output. The name phase must be cataloged in a sublibrary accessible from the VSE/POWER partition. The name can be up to eight alphanumeric (characters, numbers, and special characters). If omitted, the system default FCB is used. The default name can be specified with an asterisk.

copy The number of copies to be printed from the print queue. The value can be from 1 to 255. If omitted, VSE/POWER prints one copy. The default value is 1.

Chapter 29. Generation command and option descriptions

This chapter describes the generation commands, subcommands, parameters, and options. Options (keyword parameters beginning with a /) can be used as shown on commands or in generation options parts.

See the following chapters for command syntax diagrams:

- “Chapter 5. Command interface for C++ generation” on page 21
- “Chapter 9. Command interface for Java generation” on page 43
- “Chapter 15. Command interface for COBOL generation” on page 167
- “Chapter 19. Command interface for Web transaction program generation” on page 187
- “Chapter 22. Command interface for Java wrapper generation” on page 201

HPTCMD commands

This section describes the HPTCMD command and HPTCMD subcommands.

HPTCMD command

The **HPTCMD** command provides the command interface functions for the VisualAge Generator Developer. The first parameter you enter after the HPTCMD command is the subcommand. The command can be entered from a system prompt or from within another program or command file.

HPTCMD subcommands

GENERATE	Specifies that you want to generate a program, table, or map group. You can also specify options that affect how a part is generated.
PREPARE	Specifies that you want to run the preparation script created during the generation process. You can also specify options that affect how a part is prepared.
START	Specifies that you want to start the server process that runs HPTCMD subcommands.
STOP	Specifies that you want to stop the server process that runs HPTCMD subcommands.
VALIDATE	Specifies that you want to validate a program, table, or map group.

You can specify option values as part of the validation.

You can use this command even if you do not have the VisualAge Generator Developer installed on your machine. The VALIDATE subcommand enables you to validate your source before submitting your program for generation.

The GENERATE, PREPARE and VALIDATE subcommands use many of the same parameters and options. See the appropriate syntax diagrams to determine what parameter or option is used. Use the following information to determine the meaning of the parameter or option.

Required parameters for subcommands

The parameters listed below are required but cannot all be used at the same time. For specific syntax information, refer to the syntax diagrams listed at the beginning of this chapter.

filename

Is a positional parameter on the PREPARE subcommand which specifies the name of the preparation command file generated for a part.

The file name is the name of the part specified on the GENERATE subcommand. Specifying the file extension is optional. If an extension is not specified, PRP is assumed.

If the preparation process starts as a result of the /PREP generation option being specified, the path for the file is determined from the value specified for the /GENOUT generation option.

If the path is not explicitly specified, and the /GENOUT generation option is not specified, the DPATH directories are searched for the file.

partname

Is a positional parameter on the GENERATE command which specifies the program, table, or map group you want to generate or validate.

The generation process determines the part type from the part name.

You can specify multiple parts separated by a comma. Spaces between the part names is not allowed.

/CONFIGMAPNAME

Specifies the ENVY configuration map to be loaded to access the program, table, or map group, as well as all its associated parts during generation or validation. The value must be specified in quotation marks ("). This parameter is valid only in VisualAge Generator for Smalltalk.

/CONFIGMAPVERSION

Specifies the version of the ENVY configuration map identified by the /CONFIGMAPNAME parameter. The value must be specified in quotation marks (""). This parameter is valid only in VisualAge Generator for Smalltalk.

/PROJECT

Specifies the name and the version of a Java project. For example:

```
/PROJECT="AlphaProj", "1.2"
```

This option is repeatable.

/SYSTEM

Specifies the target system where the generated program runs.

In VisualAge Generator, if you generate the same program, map group, or table for more than one target run-time environment, you must specify a different output directory for each generation. Otherwise, the output from the later generations overlays the output from the first generation. You can specify the symbolic parameter EZEENV as one of the subdirectories on the /GENOUT option. Specifying /GENOUT causes outputs for different systems to be written to different directories.

The following table shows the batch terms, the corresponding user-interface terms, and the type of generated output for all the values that can be specified for the target system.

Table 76. The /SYSTEM generation option values

Batch	User interface	Output
AIX	AIX	C++
AIXCICS	AIX CICS	C++
HP	HP-UX	C++
IMSBMP	IMS BMP	COBOL
IMSVS	IMS VS	COBOL
JAVAWINNT	Windows NT	Java
JAVAGUI		GUI (on VisualAge Java only)
JAWRAPPER	Java Wrapper	Java (on VisualAge Java only)
MVSBATCH	MVS Batch	COBOL
MVSCICS	MVS CICS	COBOL
NTCICS	Windows NT CICS	C++
OS2CICS	OS/2 CICS	COBOL

Table 76. The /SYSTEM generation option values (continued)

Batch	User interface	Output
OS2	OS/2	C++
OS2GUI		GUI (on VisualAge Smalltalk only)
OS400	OS/400	COBOL
SCO	SCO OpenServer	C++
SOLARIS	Solaris	C++
SOLACICS	Solaris CICS	C++
TSO	MVS TSO	COBOL
VMCMS	VM CMS	COBOL
VMBATCH	VM Batch	COBOL
VSEBATCH	VSE Batch	COBOL
VSECICS	VSE CICS	COBOL
WINGUI		GUI (on VisualAge Smalltalk only)
WINNT	Windows NT	C++

The /CONFIGMAPNAME, /CONFIGMAPVERSION, and /PROJECT options require a value.

Optional parameters for subcommands

The phrase in parentheses that follows the option name is the field name that is displayed on the user interface.

/ANSISQL (ANSI SQL statements)

Specifies that you want to generate SQL statements using the ANSI SQL format.

Generate ANSI SQL statements only if you use a non-IBM database manager on MVS, VM, or VSE environments.

Do not generate ANSI SQL statements if your database manager is DB2 on MVS, DB2/VSE on VSE, DB2/400 on OS/400, or SQL/DS on VM.

ANSI SQL statements do not support the Execution Time Statement Build option for SQL processes or the specification of table names as host variables at run time. You can only use these functions with the IBM database managers.

When specifying ANSI SQL statement generation, consider the following items:

- The I/O error values (ERR, NRF, DED, UNQ, and HRD) are set by treating the ANSI SQL database manager return code as though it had the same meaning as the corresponding DB2, DB2/VSE, or SQL/DS VM return code. If the return codes have different meanings, check the value in the EZESQCOD special function word.
- Any of the preparation templates and procedures for SQL programs that might need to be tailored to use the procedures required by your ANSI SQL database manager. This includes templates and procedures that perform the following functions:
 - SQL precompile
 - Link-edit with database manager stub
 - DB2 BIND
- Any of the run-time templates that might need to be tailored to meet the requirements of your ANSI SQL database manager.

The default is /NOANSISQL.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/BIND (Bind Control)

Specifies the BIND control part to search for the program BIND command file.

A BIND control part is required only for DB2 programs generated for MVS systems. Preparation templates shipped with VisualAge Generator contain default BIND command statements used to generate a default BIND control part when a DB2 program is generated. The template used to produce the BIND control part is determined by the target environment, whether a DB2 work database is used, and whether a DL/I database is used.

To determine whether a BIND control part is to be produced from default templates or from a BIND control part, the generator searches the workspace/image into which the generation project/configuration map was loaded for a BIND control part named *prgmname.bind_value* if you specified the /BIND option or *prgmname* if you did not specify /BIND. If a matching BIND control part is found, it is used to produce the program's BIND control part. Otherwise, the program's BIND control part is produced from the appropriate template. Regardless of the input used to produce the BIND control part, the BIND command file is named *prgmname.BDC*.

We recommend that you always use the /BIND option, because it is difficult to create a bind control part with the same name as the program part.

Note: Binding a program targeted for CICS for OS/2 is accomplished by setting BINDPARAM=Y. This causes COBOL/2 to perform the SQL bind for the program.

For COBOL Users

COBOL has a limitation of 128 characters for the command line parser. Characters exceeding that limitation are truncated. Do not use names for the /GENOUT option or the /BIND option that might cause the total length of the command line to exceed this limitation.

If the limitation is exceeded and the line is truncated, any of the following might occur:

- An SQL message might be displayed if the /SQLDB option is truncated.
- A COBOL compiler message might be displayed if the /GENOUT option or the /BIND option is truncated.

There is no default value.

Table 77 on page 375 shows the generation options and the valid environments for each option.

See “Chapter 27. BIND control parts” on page 257 for more information about BIND.

/CHECKTYPE (Substructured data items)

Specifies that you want to check the structures of records and tables for possible data type conflicts that can cause run-time errors.

An example of the type of conflict this checking reports is a character (CHA) data item that is substructured by a packed (PACK) data item.

You can specify the following values:

NONE

Specify **NONE** if you do not want to check for potential conflicts in the data types of substructured data items at validation.

LOW

Specify **LOW** to check for conflicting data types in the lower-level data items only. The highest level of the structure is not checked.

For example, if a record or table is defined with the first data item at level 03 and all other data items are substructured under the first data item, then the first data item is not included in the type checking. This

helps to minimize the number of messages issued if the entire structure is defined as character so it can be moved as a block.

ALL

Specify **ALL** to check for conflicting data types in all levels of a substructured data item.

An information message is issued if a conflict is found.

Specifying a value other than NONE increases the time needed for validation.

The default value is NONE.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/CICSDBCS (CICS translator supports DBCS)

Specifies that your MVS or VSE translator supports double-byte character set (DBCS) names. Specify /CICSDBCS if your program uses DBCS part names.

If DBCS names are not supported, all DBCS names are assigned an alias name at generation.

Note: CICS/ESA 3.1 and VSE/CICS 2.2 or higher support DBCS names; earlier releases do not.

Note: All the CICS for OS/2 environments that the VisualAge Generator product supports, support DBCS characters. This option is set for the CICS for OS/2 environment.

The default is /NOCICSDBCS.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/CICSENTRIES (CICS entries)

Specifies that you want to generate CICS program and transaction definitions.

You can specify the following values:

Specify **NONE** if you do not want to generate CICS program and transaction definitions.

Specify **RDO** to generate model resource definition online (RDO) program and transaction definitions.

The RDO option can be used with CICS for AIX, CICS for Solaris, CICS for Windows NT, CICS for OS/2, CICS for VSE/ESA, and CICS for MVS/ESA systems. CICS for MVS/ESA V2 systems do not support the batch utilities for RDO.

Specify **MACRO** to generate model CICS DFHPPT and DFHPCT model statements.

The default value for `/CICSENTRIES` is NONE.

Table 77 on page 375 shows the generation options and the valid environments for each option.

See “CICS for OS/2 table entries” on page 163 for additional information on generating program and transaction definitions in CICS for OS/2.

/COMMENTLEVEL (Comment level) (COBOL)

Specifies the level of comments you want to include when you are generating a COBOL program.

See “`/COMMENTLEVEL (Generate comments) (C++)`” on page 305 if you are generating C++ programs.

Raising the comment level increases the size of the source part, but it has no effect on the size or performance of the compiled COBOL program. However, raising the comment level might significantly increase the time needed to generate the program, and it also might increase the time needed to transfer the files and compile the program.

You can select any of the following values:

0 Specify **0** if you do not want comments included in the generated program.

Note: Generation option comments are still included in the generated program for debugging purposes.

1 Specify **1** if you want only the following comments included in the generated program:

- Comments on assigned alias names
- Standard generation information (for example, the generation options and generation date and time)

2 Specify **2** if you want the comments specified for **1**, and the following comments, included in the generated program procedure division:

- Program or table prologue
- Function descriptions

- 3 Specify 3 if you want the comments specified for 1, 2, and the following comments, included in the generated program data division:
- Record prologues
 - Data item descriptions
- 4 Specify 4 if you want the comments specified for 1, 2, and 3, and all VisualAge Generator statements and comments, included in the generated program.
- This option helps in tracing the COBOL statements back to the VisualAge Generator statements and options.

The default is STATEMENTS.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/COMMENTLEVEL (Generate comments) (C++)

Specifies whether comments are included in the generated source code when you are generating C++.

See “/COMMENTLEVEL (Comment level) (COBOL)” on page 304 if you are generating COBOL programs.

Note: Some generation option comments are always included in the generated program for debugging purposes.

Approximately one brief comment is generated for each language statement.

Specify 1 if you want comments included in the generated program.

Specify 0 if you do not want comments included in the generated program.

Raising the comment level increases the size of the source part, but it has no effect on the size or performance of the compiled C++ program. However, raising the comment level might significantly increase the time needed to generate the program, and it also might increase the time needed to transfer the files and compile the program.

The default is 1.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/CONTABLE (Conversion table)

Specifies the conversion table to be used for converting the text in generated binary objects from OS/2 ASCII or Windows ASCII to the corresponding UNIX ASCII code page or to MVS, OS/400, VM, or VSE EBCDIC code pages.

The default conversion table name depends on the locale (which is formed by combining the language and territory as in 'english-nl') and the source and target environment. There is no default value if conversion is not needed. These default conversion tables are the same as those used during run time.

For information on how to define conversion tables for other environments, refer to the *VisualAge Generator Client/Server Communications Guide*.

For information on customizing VisualAge Generator to suit your national language requirements, refer to the National Language Support section in the *VisualAge Generator Installation Guide*.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/CREATEDDS (Create DDS files)

Specifies that OS/400 environment data description specification (DDS) files are to be created from the record definitions used for file I/O operations associated with the program being generated.

The default value is /NOCREATEDDS.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/CURRENCY (Currency Symbol)

Specifies a currency symbol comprising one to three characters. If you do not specify this option, the default value is derived from the system locale.

To specify a character that is not on your keyboard, hold down the Alt key and use the numeric key pad to type the character's decimal code. On Windows NT the decimal code for the Euro is 0128; on OS/2 it is 0213.

/DATA (Data)

Specifies whether you want VisualAge Generator Server for MVS, VSE, and VM to allocate its working storage with 24- or 31-bit addresses.

This option also controls whether you want the generated preparation JCL to use the DATA(24) or DATA(31) compile options for MVS COBOL and VSE COBOL programs or whether you want the generated preparation script to use the DATA(24) or DATA(31) for VM COBOL programs.

You can specify the following values:

24

Specify **24** if you want VisualAge Generator Server for MVS, VSE, and VM to allocate its working storage with 24-bit addresses.

For CICS for MVS/ESA and CICS for VSE/ESA, you can only specify 24-bit addressing if the dynamic storage required by the generated COBOL program, mapping services program, or table is less than 64KB.

You must specify 24 bit for the following:

- Any program that calls a program linked AMODE(24)
- The first program in the run unit if any generated program in the run unit is linked AMODE(24) or if any program that uses DL/I is generated with /DATA=24
- Tables and map groups if any program that uses the table is linked AMODE(24)
- Any DL/I program in VSE batch, or in non-CICS environments on MVS if IMS/ESA is not installed

31

Specify **31** if you want VisualAge Generator Server for MVS, VSE, and VM to allocate its working storage with 31-bit addresses.

If you specify 31-bit addressing for DL/I programs in the MVS batch or MVS/TSO environments, you must install IMS/ESA and you must specify the VisualAge Generator Server for MVS, VSE, and VM IMSESA installation option as IMSESA='Y'. Refer to the Program Directory and the *VisualAge Generator Server Guide for MVS, VSE, and VM* for information on specifying the IMSESA installation option.

The default for non-CICS environments is /DATA=24. The default for CICS environments is /DATA=31.

/DBMS (Database management system)

Specifies the database management system to use for SQL programs. You can specify one of the following values:

- | | |
|---------------|--------------------------------------------|
| DB2 | generates native DB2 embedded SQL calls |
| Oracle | generates native Oracle embedded SQL calls |
| ODBC | generates dynamic ODBC SQL calls |

If the `/DBMS` option is not specified, the Database Management system value specified in the VisualAge Generator SQL Preferences is used. You can also select **Database management system** on the Validation tab of the Generation Options notebook.

There is no default value.

Table 77 on page 375 shows the generation options and the valid environments for each option.

`/DBPASSWORD (Password)`

Specifies the password of the USERID that will be connected to the database on the target machine.

You must specify a password to use with the USERID for the database.

This option is case sensitive.

There is no default value.

Table 77 on page 375 shows the generation options and the valid environments for each option.

`/DBUSER (User ID)`

Specifies the USERID that will be used to connect to the database on the target machine.

This USERID is used to connect to the database on the target machine. The USERID must be identified to the database, and have the appropriate access privileges.

This option is case sensitive.

There is no default value.

Table 77 on page 375 shows the generation options and the valid environments for each option.

`/DEBUGTRACE (Debug trace information)`

Specifies `/DEBUGTRACE` to enable tracing of the generation process.

Trace information is written to a file named *prgmname*.TRC in the directory specified by the `/GENOUT` generation option.

Use tracing only when you are providing debugging information to IBM service personnel.

The default is /NODEBUGTRACE.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/DESTACCOUNT (Account)

Specifies the name of the account on the machine the generated code and auxiliary supporting files will be sent to when transferring files using TCP/IP.

There is no default value.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/DESTDIR (Directory)

Specifies the name of the directory on a machine that generated server code and auxiliary supporting files will be sent to.

The target directory must exist. The user specified by the /DESTUID generation option must have the authority to write to this directory.

For UNIX environments, do the following:

- Specify the fully qualified path name.
- Enclose the specified value in single quotation marks (').
- Validate the case used. The environments are case sensitive.
- Be sure to use slashes (/).

For other environments, specify the directory name, not the absolute path name.

You cannot use any platform-specific keywords, for example:

- \$HOME
- tilde (~)

For Java, server code is sent to /DESTUID and gateway servlet code is sent to /JAVADESTDIR.

Example: for UNIX environments

```
'/home/myid/mysource'
```

There is no default value.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/DESTHOST (Name)

Specifies the name or numeric TCP/IP address of the target machine on which the target directory resides. You can enter up to 64 characters for the name or address. If developing on Windows NT, you must specify a fully qualified name.

You can type up to 64 characters for the name or address. If developing on Windows NT, you must specify a fully qualified name.

The UNIX environments are case sensitive. Verify the case of the value you specify for this option.

Example:

sms.raleigh.ibm.com

or

9.67.226.22

There is no default value.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/DESTLIB (Target library)

Specifies the 1- to 10-character OS/400 library name where objects created during generation are placed when transferred by the preparation process.

The default is QGPL.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/DESPASSWORD (Password)

Specifies the password of the USERID that will be logged on to the target machine.

The password is used for the user specified by the /DESTUID generation option. The AIX, HP-UX, SCO, and Solaris environments are case sensitive. Verify the case of the value you specify for this option.

There is no default value.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/DESTUID (User ID)

Specifies the USERID that will be logged on to the target machine specified by the /DESTHOST generation option.

The UNIX environments are case sensitive. Verify the case of the value you specify for this option.

Example:

```
/DESTUID=myuserid
```

There is no default value.

/DXFRCANCEL (Cancel program after DXFR)

Specifies that you want to remove the COBOL program from memory for any program that was transferred to using a DXFRLINK with a linktype of DYNAMIC when execution of that program is complete. A COBOL CANCEL command appears in the generated COBOL of the program performing the DXFR.

The default is /NODXFRCANCEL

Table 77 on page 375 shows the generation options and the valid environments for each option.

/DXFRXCTL (Implement DXFR as an XCTL)

Specifies that you want to use an XCTL command to implement a DXFR statement. This value changes the default DYNAMIC linktype on a DXFRLINK to XCTL.

After specifying this value, by default, all programs will perform an XCTL when a DXFR is requested. You can override the default with :DXFRLINK linkage table entries. Because XCTL is already the default for CICS environments, this option has no effect there.

Note that this option results in slower performance because the run-time environment must be reloaded each time a DXFR is performed.

The default is /NODXFRXCTL.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/EJBGROUP (Enterprise Java Bean Group)

Specifies that you want to create a session bean assigned to a particular Enterprise Java Bean (EJB) group.

The group must already exist.

By default VisualAge Generator creates Java wrappers (one for the program and one for each record parameter) but does not create a session bean. For more information about Java wrappers, see VisualAge Generator Client/Server Communications Guide.

This option is valid only with the Java Wrapper target.

/ENDCOMMAREA (End COMMAREA with FFFF)

Specifies /ENDCOMMAREA if you want a full word containing X'FFFFFFFF' appended to the end of the COMMAREA on calls to non-VisualAge Generator programs. The length of the COMMAREA is increased by 4 bytes.

COMMPTR is the default parameter format for all CICS environments.

The default is /NOENDCOMMAREA.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/ERRDEST (Error destination)

Specifies the destination of the diagnostic messages in the IMS/VS and IMS BMP environments.

The error destination is specified as one of the following:

- The IMS transaction code associated with a BMP program that lists error diagnostics associated with run-time errors
- The logical terminal identifier associated with the printer where the diagnostics are sent

This option is effective at run time only if the option is specified for the first program in the run unit.

The default for IMS/VS is ELADIAG, a transaction code for a BMP utility provided with the VisualAge Generator Server for MVS, VSE, and VM function that prints the diagnostics. The default for IMS BMP is to write the messages to the file ELAPRINT instead of a message queue.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/FASTPATH (Run as a fast-path program)

Specifies /FASTPATH if you want the program to run as an IMS fast-path program. If the /FASTPATH option is specified and the run unit does not end successfully, VisualAge Generator Server for MVS, VSE, and VM issues a 1602 abend.

If the /NOFASTPATH option is specified and the run unit does not end successfully, VisualAge Generator Server for MVS, VSE, and VM issues a ROLL request.

This option is effective at run time only if the option is specified for the first program in the run unit.

The default is /NOFASTPATH.

Table 77 on page 375 shows the generation options and the valid environments for each option.

Refer to the *VisualAge Generator Design Guide* for detailed information on IMS fast-path programs.

/FOLD (Fold to uppercase)

Specifies that you want folding to uppercase to occur for the output.

When you select folding, the source statements and comments are transformed from lowercase to uppercase characters in printed reports and in the generated programs where the source statements and comments appear as COBOL comments.

The default is /NOFOLD.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/FTPTRANSLATIONCMDDBCS (FTP DBCS Translation Command)

Specifies that the DBCS translation tables must be set for the remote system when transferring files using FTP. This is normally not necessary, because any needed translation is done automatically on the local system.

There is no default value.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/FTPTRANSLATIONCMDSBCS (FTP SBCS Translation Command)

Specifies that the SBCS translation tables must be set for the remote system when transferring files using FTP. This is normally not necessary, because any needed translation is done automatically on the local system.

There is no default value.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/GENAUTHORTIMEVALUES

Adds code to UI record JSPs and UI record beans to let Web page designers quickly view simulated results of changes to generated JSPs. The added code does not affect results when the JSP is invoked from the Web transaction gateway.

A method, `initAuthorTimeValues()`, is added to UI record beans to initialize each UI record data item that the UI record JSP uses to produce an HTML element. The initialization values are determined only by the UI type and the data type of the data items. For example, a data item with a UI type of OUTPUT and a data type of NUM with two decimal places is assigned a value of 99.99. A scriptlet is added to the JSP to invoke the `initAuthorTimeValues()` method.

When you use a URL in a browser to invoke the JSP directly rather than by starting a Web transaction from the gateway, `initAuthorTimeValues()` is invoked to initialize data items in the UI record rather than using values produced in the Web transactions. These values are displayed formatted by the JSP just as if they were set by a Web transaction, but you do not need to set up the actual Web transaction on your application server to see results of JSP modifications. Although you cannot use this methodology to test interaction between Web pages, links will be properly displayed. You can modify the `initAuthorTimeValues()` method to produce more realistic default data.

The default is `/NOGENAUTHORTIMEVALUES`.

/GENHELPMAPS (Help map group)

Specifies that you want to generate a help map group with the program.

Specify `/NOGENHELPMAPS` if you do not want to generate a help map group.

This option is available only if you are generating a program.

The default is `/GENHELPMAPS`.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/GENMAPS (Map group)

Specifies that you want to generate a map group with the program.

Specify `/NOGENMAPS` if you do not want to generate a map group.

This option is available only if you are generating a program.

The default is `/GENMAPS`.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/GENOUT (Generated output directory)

Specifies a directory on your workstation where you store the generation outputs. When you specify the directory path for the `/GENOUT` option, you can use symbolic parameters for subdirectory names.

Notes:

- If you specify this option in a command file and you want to use a symbolic parameter as part of the value for this option, you must specify two percent signs (`%%`) as the symbolic parameter delimiter.
- If you specify this option on the command line or in a generation options file, the symbolic parameter delimiter is one percent sign (`%`).
- The user interface always places the options in a command file. If you use symbolic parameters in values specified for the interactive interface, always use two percent signs as the symbolic parameter delimiter. For example, to use the symbolic parameter `EZEENV`, specify `%%EZEENV%%`. This ensures that the command interpreter correctly replaces the symbolic parameter value.

When you are generating for an AIX target environment, the generated output is first created in this directory. The generated output is then transferred to the directory specified by the `/DESTDIR` option.

For COBOL Users

COBOL has a limitation of 128 characters for the command line parser. Characters exceeding that limitation are truncated. Do not use names for the `/GENOUT` option or the `/BIND` option that might cause the total length of the command line to exceed this limitation.

If the limitation is exceeded and the line is truncated, any of the following might occur:

- An SQL message might be displayed if the /SQLDB option is truncated.
- A COBOL compiler message might be displayed if the /GENOUT option or the /BIND option is truncated.

Example:

The following is an example of using the /GENOUT generation option at the command prompt or in an options file:

```
/GENOUT=C:\Program Files\VAST\%EZEUSRID%\%EZEENV%
```

The following is an example of using the /GENOUT generation option in a command file:

```
/GENOUT=C:\Program Files\VAST\%%EZEUSRID%\%%EZEENV%
```

The values of the symbolic parameters replace the symbols in the path name when the generation process writes out a generated object, if you type the following:

```
C:\Program Files\VAST\%%EZEUSRID%\%%EZEENV%
```

In the above example, if EZEUSRID is set to PROJECT1 and the target system for generation is IMSVS, then the generated objects are written to directory C:\Program Files\VAST\PROJECT1\IMSVS.

If the specified directory does not exist, it is created.

The default directory is the directory where the server process is running.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/GENPROPERTIES

Specifies that you want to generate with your Java server program a properties file that contains settings derived from the generation options file, linkage table, and resource associations file.

The generated properties file is given the same name as the generated part with a `.properties` file extension and is saved in the same directory with your generated Java code. The properties file is saved as text so you can open it using a text editor and further customize it.

Properties shown in the file include:

- `vgj.nls.code` (from /TARGNLS)

- `vgj.datemask.gregorian.long.NLS` (where NLS is the language code specified in preferences)
- `vgj.datemask.julian.long.NLS` (where NLS is the language code specified in preferences)
- `vgj.jdbc.default.database` (from `/SQLDB`)
- `vgj.jdbc.default.database.userid` (from `/SQLID`)
- `vgj.jdbc.default.database.user.password` (from `/SQLPASSWORD`)
- linkage properties (from `/LINKAGE`)
- resource association properties (from `/RESOURCE`)

By default, no properties file is generated.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/GENRESOURCEBUNDLE (Generate as resource bundle)

Specifies generation of Java resource bundles.

Java resource bundles are generated in the following cases::

- You are generating only a UI record or message table and have specified option `GENRESOURCEBUNDLE`.
- You are generating a web transaction; in this case, if you have specified option `GENRESOURCEBUNDLE`, one resource bundle is generated for each UI record and one for each message table; but if you have not specified option `GENRESOURCEBUNDLE`, a resource bundle is generated only for the message table.

UI record

If `/GENRESOURCEBUNDLE` is specified, labels, titles, and help text that might be presented to a user are generated into a resource bundle that can be translated. If `/GENRESOURCEBUNDLE` is not specified, labels, titles, and help text are generated into the bean.

UI records — Web transactions

If `/GENRESOURCEBUNDLE` is specified, labels, titles, and help text for each UI record used in the Web transaction are generated into separate resource bundles. Otherwise, they are generated into the beans.

The name of the resource bundle generated for a UI record is as follows:

```
NameRBund1e_JavaLocale.java
```

Name is the message table prefix. See the `/RESOURCEBUNDLELOCALE` option for a description of *JavaLocale*.

Table If /GENRESOURCEBUNDLE is specified, the table will be generated as a resource bundle that contains message keys and associated message text.

In order to be used by a UI record bean, the table must be named *PrefixNLSValue* where *Prefix* is the message table prefix in the bean and *NLSValue* is /TARGNLS. The table must be defined as follows:

column 1	Num4, Char, or Mixed, 1-254
column 2	Char or Mixed, 1-254

Tables — Web transactions

For a Web transaction, the message table is always generated as a Java resource bundle. The message table prefix and /TARGNLS are used to determine which message table to use.

The name of the resource bundle generated for table *PrefixNLSValue* is as follows:

*Prefix*RBundl*e_**JavaLocale*.java

Prefix is the message table prefix. See the /RESOURCEBUNDLELOCALE option for a description of *JavaLocale*.

The name of the table should have a 1-4 character prefix followed by a 3-character NLS value, and the NLS value should be matched at generation time by the setting of generation option TARGNLS.

This option is only available if you are generating a UI record, table or Web Transaction program..

The default is /NOGENRESOURCEBUNDLE.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/GENRET (Issue RETURN IMMEDIATE)

Specifies that you want CICS to issue a RETURN IMMEDIATE.

The default is /NOGENRET. For more information about the default generated code, see the *VisualAge Generator Design Guide*.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/GENTABLES (Tables)

Specifies that you want to generate all the tables associated with the program being generated.

Specify `/NOGENTABLES` if you do not want to generate the tables.

For user message tables, only the message table for the target NLS is generated.

This option is available only if you are generating a program.

The default is `/GENTABLES`.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/GENUIRECORDS

Specifies that the user interface records in the program are to be generated.

The following Java files can be generated for a user interface record:

- UI record bean
- UI record object
- Resource bundle
- Edit table definition
- Edit table contents
- Edit function local-storage record definition
- Java Server Page (JSP)

This option is available only if you are generating a Web transaction program.

The default is `/GENUIRECORDS`.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/GROUPNAME (Group name)

Specifies the 1- to 8-character group name to which this resource belongs.

Type a valid 1- to 8-character group name.

Resource management is simplified when related transactions and resources are associated with groups.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/INEDIT (Input edit)

Specifies whether you want to do input editing only for modified fields on a map.

This option affects the operation of the TEST map-item statement in the program.

Valid values are ALL and INONLY. The default is ALL.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/INITADDWS (Initialize additional working storage records)

Specifies that you want to initialize working storage records in the table and additional records list of a program.

The primary working storage record specified for the program is always initialized. Specify /NOINITADDWS if you do not want to initialize these working storage records.

The following statements are true if you specify /INITADDWS:

- Records are initialized as though a SET record EMPTY statement is specified at the beginning of the program you are generating. This means that you do not need to code a SET record EMPTY statement for each working storage record.
- Some abends caused by data that has not been initialized can be avoided.

Note: Records received as parameters in a called program and redefined records are never initialized.

The following is true if you specify /NOINITADDWS:

- If your program has a large number of records with many fields that do not require initialization, performance might be slightly better.
- If your program already uses SET record EMPTY to initialize your working storage records, using this option eliminates additional initialization of the working storage records, causing improved performance. The change in performance is more noticeable for repeatedly called programs.

The default is /INITADDWS.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/INITRECD (Initialize records)

Specifies that you want to initialize serial, indexed, relative, SQL row, and DL/I segment records when the program starts to run.

Specify `/NOINITRECD` if you do not want to initialize these records.

The following statements are true if you specify `/INITRECD`:

- Records are initialized as though a SET record EMPTY statement was specified at the beginning of the program you are generating. This means that you do not need to code a SET record EMPTY statement for each record.
- Some abends caused by data that has not been initialized can be avoided.

Note: Records received as parameters in a called program and redefined records are not affected by this option.

The following statements are true if you specify `/NOINITRECD`:

- If your program has a large number of records with many fields that do not require initialization, performance might be slightly better.
- If your program already uses SET record EMPTY to initialize your non-working storage records, using this option eliminates additional initialization of the non-working storage records, causing improved performance. The change in performance is more noticeable for repeatedly called programs.

The default is `/INITRECD`.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/JVADESTDIR (Java directory)

Specifies the name of the directory on a target machine that generated gateway Java gateway code and auxiliary supporting files will be sent to.

The target directory must exist. The user specified by the `/JVADESTUID` generation option must have the authority to write to this directory.

For UNIX environments, do the following:

- Specify the fully qualified path name.
- Enclose the specified value in single quotation marks (').
- Validate the case used. The environments are case sensitive.
- Be sure to use slashes (/).

For other environments, specify the directory name, not the absolute path name.

You cannot use any platform-specific keywords, for example:

- \$HOME
- tilde (~)

For Java, server code is sent to /DESTUID and gateway servlet code is sent to /JVADESTDIR.

Example: for UNIX environments

```
'/home/myid/mysource'
```

There is no default value.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/JVADESTHOST (Name)

Specifies the name or numeric TCP/IP address of the gateway target machine on which the Java target directory resides. You can enter up to 64 characters for the name or address. If developing on Windows NT, you must specify a fully qualified name.

You can type up to 64 characters for the name or address. If developing on Windows NT, you must specify a fully qualified name.

The UNIX environments are case sensitive. Verify the case of the value you specify for this option.

Example:

```
sms.raleigh.ibm.com
```

or

```
9.67.226.22
```

There is no default value.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/JVADESTPASSWORD (Password)

Specifies the password of the USERID that will be logged on to the gateway target machine.

The password is used for the user specified by the /JVADESTUID generation option. The AIX and Solaris environments are case sensitive. Verify the case of the value you specify for this option.

There is no default value.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/JVADESTUID (User ID)

Specifies the USERID that will be logged on to the gateway target machine specified by the /DESTHOST generation option.

The UNIX environments are case sensitive. Verify the case of the value you specify for this option.

Example:

```
/JVADESTUID=myuserid
```

There is no default value.

/JAVASYSTEM (Java target system)

Specifies the type of target system for Java gateway output. This option corresponds to the gateway target system you can select on the Generate window. This system can be AIX, OS2, OS390, OS400, SOLARIS, or WINNT.

/JAVASYSTEM must be specified for preparation files to be generated for the gateway.

There is no default value.

/JOB CARD (JOB card)

Identifies the name of the file containing the job statement you want to use for generated preparation or run-time JCL or CL

The file name must be specified in one of the following ways:

- A fully qualified file name. Symbolic parameters are permitted in the directory specification. See “Chapter 12. Symbolic parameters” on page 123 for more information about symbolic parameters.
- The name of a file located in a directory specified in the DPATH environment variable of the server process. See “Chapter 15. Command interface for COBOL generation” on page 167 for more information about the server process.

The default file name for MVS is EFK2MJOB.TPL.

The default file name for VSE is EFK2VJOB.TPL.

Table 77 on page 375 shows the generation options and the valid environments for each option.

See “JOB statements” on page 74 for more information about the job statement.

/JOBNAME (Job name)

Specifies the value used for job name in the default job statement built for the preparation of run-time JCL or CL batch job.

The name must be a valid MVS or VSE job name.

The job name is used to replace the symbolic parameter, EZEJOB, in the job template. If you modify your VSE templates to use the symbolic parameter, EZEJOB, this option can be used in VSE environments.

The default name is built using the value specified in the .ini file for the Options/Preferences page. If that value is not set, the value of the EZEUSRID environment variable is concatenated with the letter A. Because the job name value is limited to a length of 8 characters, only the first 7 characters are used from EZERUSRID values that are longer than 7 characters.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/JSPRELDIR

Specifies a path, relative to the JSP directory, where its associated beans are stored.

The relative path is a partially qualified uniform resource locator (URL). Using this option allows you to place beans in directories other than the root of the JSP you are using.

The URL must be enclosed in quotes so that the forward slashes are interpreted correctly.

There is no default for this option and a value is not required.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/LEFTJUST (Left justify)

Specifies that you want CHA, MIX, and DBCS map variables defined with JUSTIFY LEF justification on output.

Specify /NOLEFTJUST if you do not want data left justified.

Specifying /LEFTJUST eliminates leading spaces for map variables defined with JUSTIFY LEF. It is safe to request left justification if the values displayed are always left justified on input or are created by the program.

Not requesting left justification might provide better performance and smaller load module size.

Note: This option has no effect on justification of input data; this option affects only the generation of map groups that contain printer maps. It has no effect on the generation of programs, tables, or map groups that only contain terminal maps.

The default is /LEFTJUST.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/LINEINFO (Line trace information)

Specifies that you want to include line information in columns 73–80 of the generated COBOL output.

Line information identifies which generation function produced the output line. You might have better performance if you do not request line information.

The following statements are true when you specify /LINEINFO:

- The size of the load module is not affected.
- The time needed to transfer the COBOL source is increased.

The default is /NOLINEINFO.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/LINES (Lines per page)

Specifies the number of lines you want to print on each page in the generation listing file.

You can enter one of the following values:

- 0** Specifies that no pages breaks will be inserted.
- number* Specifies the number of lines (20-999) per page.

The default is 55.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/LINKAGE (Linkage table)

Specifies the part name where the linkage table information is stored.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/LINKEDIT (Link edit)

Specifies the part name of the optional link edit part, which contains the linkage editor control statements for link-editing the program with other programs for the MVS, VSE, or VM environments.

Table 77 on page 375 shows the generation options and the valid environments for each option.

See “Link edit parts” on page 53 for more information about linkage editor control statements.

/LISTING /LISTINGONERROR, /NOLISTING (Generation listing)

Specifies whether a generated output listing will be created. The following options specify under what conditions a listing is generated:

/NOLISTING

Specify this option if you do not want an output listing generated.

/LISTING

Specify this option if you want a listing of the part source code generated.

/LISTINGONERROR

Specify this option if you want a listing of the part source code to be generated only if any error messages are issued.

The generated output listing includes the generation options, resource associations, and linkage table entries used to generate the part source code.

The default is /NOLISTING.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/LISTINGONERROR

See /LISTING on page “/LISTING /LISTINGONERROR, /NOLISTING (Generation listing)”.

/LOCVALID (Local data items)

Specifies that you want local data items validated by comparing each local data item definition with the global item definition of the same name, if one exists. If you request validation and the definitions are different, a message is issued and generation continues.

You need not specify the /NOLOCVALID option if you do not want local data item definitions validated; /NOLOCVALID is the default.

When you request local data item validation, it takes longer for validation to complete.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/LOG (Log identifier)

Specifies that you want the run-time errors detected by VisualAge Generator Server for MVS, VSE, and VM written to the IMS system log.

You need not specify /NOLOG if you do not want IMS log records written to the IMS system log; /NOLOG is the default. Messages written to the IMS log are identical to messages that are written to ERRDEST.

If you specify that you want messages logged, you must specify a log identifier. The log identifier is the IMS log record identifier that is used when run time error messages are written to the IMS log. The identifier is a 2-digit hexadecimal number ranging from A0 through FF.

This option is effective at run time only if the option is specified for the first program in the run unit.

The Table 77 on page 375 shows the generation options and the valid environments for each option.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/MATH (Math)

Specifies whether you want the VisualAge Generator calculation method of truncating intermediate results emulated in the COBOL program.

The truncation method used is significant if you are converting an existing CSP/AE program and you want to ensure that the results are consistent with the original program.

You can specify the following values:

COBOL

Select **COBOL** to use COBOL truncation algorithms.

COBOL might provide faster performance, smaller load module size, and better accuracy.

CSPAЕ

Select **CSP/AE** if you want truncation of intermediate results in calculations to the same number of significant digits as the result field. This is the truncation algorithm used by the CSP/AE product. Specify this option only for programs originally developed to run with CSP/AE that have code that is dependent on this truncation algorithm.

Note: The VisualAge Generator test facility supports only the COBOL method of truncation.

The default is COBOL.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/MFSDEV

Specifies the devices for which maps are to be generated and the parameters to be used on the MFSDEV statement.

Each set of device options specifies a terminal or printer device type and the MFS parameters that correspond to that device.

Note: You Can specify the /MFSDEV generation option on the GENERATE subcommand only or in a generation options file. The VisualAge Generator Developer interactive interface has no corresponding entry for the /MFSDEV generation option.

You can specify the following values:

ad device

Specifies a valid device type that can be specified during map definition.

Enclose the value in single quotation marks (') or double quotation marks ("). For additional information on the valid device types that can be specified, refer to the VisualAge Generator Developer online help system.

mfs device

Specifies one or more of the parameters used on a DEV statement for MFS.

Enclose the string of parameters in single quotation marks (') or double quotation marks ("). The information in the MFS device variable is based on the information supplied for the TERMINAL and TYPE macros in your IMS system definition.

EATTR, NCD, or NOEATTR

Indicates whether the device supports extended attributes, and whether a CD (color default) extended attribute is generated for map fields defined as MONO. These values are optional and allow you to override the default value specified using the /MFSEATTR, /MFSEATTRNCD, or /NOMFSEATTR generation option.

- IMS/V5
- IMS BMP

Refer to the *VisualAge Generator Server Guide for MVS, VSE, and VM* for details about setting this attribute if you select additional devices.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/MFSEATTR (MFS extended attribute)

Specifies whether you want to include extended attribute support in the generated MFS.

The default is /MFSEATTR.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/MFSEATTRNCD

See /MFSEATTR “/MFSEATTR (MFS extended attribute)”.

/MFSIGNORE (Include IGNORE for SOR)

Specifies that you want to include IGNORE for the SOR parameter on the message statement for the MID and MOD in the generated message format service (MFS).

Specify /MFSIGNORE only if the /MFSDEV option specifies FEAT=IGNORE for all of the devices used by the map group you are generating.

The default is /NOMFSIGNORE.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/MFSTEST (Use test library)

Specifies that you want the preparation JCL to place MFS control blocks in a test library.

You need not specify `/NOMFSTEST` if you want the preparation JCL to place MFS control blocks in a staged FORMAT library; `/NOMFSTEST` is the default.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/MSGTABLEPREFIX

Specifies a message table prefix similar to the message table prefix defined in a program. This is a one- to four-character prefix of the message table that contains the keys used by user interface (UI) records.

This option is available only if you are generating a UI record. If you are generating a Web transaction program, the message table prefix defined in the program is used.

The default is no prefix.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/MSP (Mapping service program)

Specifies the type of print services programs you want to generate for IMS BMP or MVS batch. For MVS/TSO, VM batch, VM CMS, and VSE batch the print service program is always SEQ.

You can specify the following values:

ALL

Specify **ALL** to generate all the types of mapping services programs that can be used in the environment (GSAM, MFS, and SEQ, for IMS BMP; GSAM and SEQ for MVS batch). The default value is **ALL** in the IMS BMP environment but not in the MVS batch environment.

GSAM

Specify **GSAM** to generate a batch mapping services program that supports writing to a GSAM file. **GSAM** is available only for the IMS BMP and MVS batch target systems.

MFS

Specify **MFS** to generate a message format service (MFS) print mapping services program and the MFS source definitions. If the map

group contains terminal maps, the COBOL copybooks for MFS and a map group format module are also generated. This option is valid only for IMS BMP.

SEQ

Specify **SEQ** to generate a batch mapping services program for IMS, MVS, VM, or VSE that supports writing to a printer or QSAM file. This is the default for MVS batch and the only option for MVS/TSO, VM batch, VM CMS, and VSE batch.

You can enter multiple values separated by commas (.). For example, for IMS BMP, /MSP=MFS,SEQ,GSAM is valid and has the same result as specifying /MSP=ALL.

The MFS print services program has the same name as the map group. The SEQ and GSAM services, if generated, are included in the batch print services program. The batch print services program has the same name as the map group with P1 appended to the name.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/NOANSISQL

See “/ANSISQL (ANSI SQL statements)” on page 300.

/NOCICSDBCS

See “/CICSDBCS (CICS translator supports DBCS)” on page 303.

/NOCREATEDDS

See “/CREATEDDS (Create DDS files)” on page 306.

/NODXFRCANCEL

See “/DXFRCANCEL (Cancel program after DXFR)” on page 311.

/NODXFRXCTL

See “/DXFRXCTL (Implement DXFR as an XCTL)” on page 311.

/NODEBUGTRACE

See “/DEBUGTRACE (Debug trace information)” on page 308.

/NOENDCOMMAREA

See “/ENDCOMMAREA (End COMMAREA with FFFF)” on page 312.

/NOFASTPATH

See “/FASTPATH (Run as a fast-path program)” on page 313.

/NOFOLD

See “/FOLD (Fold to uppercase)” on page 313.

/NOGENAUTHORTIMEVALUES

See “/GENAUTHORTIMEVALUES” on page 314.

/NOGENHELPMAPS

See “/GENHELPMAPS (Help map group)” on page 314.

/NOGENMAPS

See “/GENMAPS (Map group)” on page 315.

/NOGENPROPERTIES

See “/GENPROPERTIES” on page 316.

/NOGENRESOURCEBUNDLE

See “/GENRESOURCEBUNDLE (Generate as resource bundle)” on page 317.

/NOGENRET

See “/GENRET (Issue RETURN IMMEDIATE)” on page 318.

/NOGENTABLES

See “/GENTABLES (Tables)” on page 319.

/NOGENUIRECORDS

See “/GENUIRECORDS” on page 319.

/NOINITADDWS

See “/INITADDWS (Initialize additional working storage records)” on page 320.

/NOINITRECD

See “/INITRECD (Initialize records)” on page 321.

/NOLEFTJUST

See “/LEFTJUST (Left justify)” on page 324.

/NOLINEINFO

See “/LINEINFO (Line trace information)” on page 325.

/NOLISTING, /LISTING, /LISTINGONERROR

See “/LISTING /LISTINGONERROR, /NOLISTING (Generation listing)” on page 326.

/NOLOCVALID

See “/LOCVALID (Local data items)” on page 327.

/NOLOG

See “/LOG (Log identifier)” on page 327.

/NOMFSEATTR

See “/MFSEATTR (MFS extended attribute)” on page 329.

/NOMFSIGNORE

See “/MFSIGNORE (Include IGNORE for SOR)” on page 329.

/NOMFSTEST

See “/MFSTEST (Use test library)” on page 330.

/NONULLFILL

See “/NULLFILL (Fill map field)” on page 334.

/NONUMOVFL

See “/NUMOVFL (Numeric overflow)” on page 334.

/NOPREP

See “/PREP (Start preparation command file)” on page 335.

/NOPREPROFILE

See “/PREPROFILE (Create preparation command file)” on page 336.

/NORECOVERY

See “/RECOVERY (Recover current error message)” on page 337.

/NORUNFILE

See “/RUNFILE (Create sample run-time JCL, Create a sample clist, or Create a sample REXX exec)” on page 342.

/NOSETFULL

See “/SETFULL (Set map item FULL)” on page 342.

/NOSPZERO

See “/SPZERO (Interpret spaces as zero in NUM and NUMC data items)” on page 343 .

/NOSQLVALID

See “/SQLVALID (SQL statements)” on page 345.

/NOSYNCDXFR

See “/SYNCDXFR (Set sync points for DXFRs)” on page 347.

/NOSYNCFER

See “/SYNCFER (Set sync points for XFERs)” on page 347.

/NOUNLOAD

See “/UNLOAD (Unload parts)” on page 353.

/NOSYSCODES

See “/SYSCODES (Use system return codes)” on page 348.

/NULLFILL (Fill map field)

Specifies that you want CHA, MIX, DBCS, and NUM map variables with FILL N (null) specified filled with null characters.

Specify /NONULLFILL if you want these map variables filled with spaces instead of null characters.

Note: This option affects only the generation of map groups that contain printer maps. It has no effect on the generation of programs, tables, or map groups that contain only terminal maps.

The default is /NULLFILL.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/NUMOVFL (Numeric overflow)

Specifies that you want to support numeric overflow checking.

Specify /NONUMOVFL if you do not want to support this checking.

If you specify /NONUMOVFL, the EZEORDER function is ignored. Division by zero results in an abend with a message. In other overflow conditions, the result is truncated causing the significant digits to be lost; there is no indication that truncation has occurred. Design your program to ensure that overflow conditions do not occur.

Specifying /NONUMOVFL, might result in smaller load modules with better performance.

The default is /NUMOVFL.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/OPTIONS (Generation options)

Specifies a part that contains additional generation options that you want applied to this generation.

The /OPTIONS option can also be specified within options parts. If the same option appears in multiple options parts, the first-found occurrence of the option applies. The exception is for options that are specified in the default generation options file with the NOOVERRIDE parameter. These options cannot be overridden.

When you use VisualAge Generator, the generations options defaults file is specified by the *defaultGenerationOptions* key in the *hpt.ini* file.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/PACKAGENAME (Package Name)

Specifies the name of a Java package in which to group related classes and interfaces.

During preparation of Java server or gateway files, a directory with the same name as the package is created for you. If necessary, a subdirectory on your target machine is also created as a subdirectory of the /GENOUT, /DESTDIR, or /JAVADESTDIR directories. Java files are then copied to this directory and compiled.

A Java package name comprises a sequence of identifiers separated by periods (.), for example, com.ibm.vgj.ugs. The default package name is my.pkg.

/POSSIGN (Positive Sign Indicator)

Specifies the character the compiler uses as the positive sign for all zoned and packed numeric data.

You can specify the following values:

F

Select **F** if you want to use F as the positive sign for all zoned and packed numeric data. This includes NUM, NUMC, PACK and PACKF VisualAge Generator data types.

C

Select **C** if you want to use C as the positive sign for all zoned and packed numeric data. This includes NUM, NUMC, PACK and PACKF VisualAge Generator data types.

If more occurrences of NUMC and PACK data types are compared to NUM and PACKF data types, then the performance is improved by using /POSSIGN=C (set **Positive sign indicator** to C).

The default value for /POSSIGN is F.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/PREP (Start preparation command file)

Specifies that, upon successful completion of generation (return code <= 4), preparation of the generated objects is automatically initiated.

Specify `/NOPREP` if you do not want preparation to start automatically. If preparation does not start automatically, the `PREPARE` subcommand must be used to start preparation. Preparation messages are written to the `PRPOUT.LOG` file in `/GENOUT`.

The default is `/PREP`.

If `/PREP` is specified, the `/NOPREPFIL` option is ignored.

Table 77 on page 375 shows the generation options and the valid environments for each option.

`/PREPFIL` (Create preparation command file)

Specifies that you want to create a preparation script at generation.

Specify `/NOPREPFIL` if you do not want to create a preparation script at generation.

If you specify `/NOPREPFIL`, you are responsible for the preparation of the generated objects on the target run-time system.

The name of the preparation script created is *partname*.PRP.

For MVS and VSE environments, a file containing the preparation JCL is created at generation. The name of the file is *partname*.JCL. For CICS for OS/2 and VM environments, a file containing the preparation REXX is created at generation. The name of the preparation REXX file is *partname*.RXP. For OS/400, the file containing the preparation REXX created at generation is called *partname*.PRP. In addition, a file containing the OS/400 preparation CL is created at generation called *partname*.CLP, and a file containing the OS/400 job stream is created at generation called *partname*.CLJ.

The Table 77 on page 375 shows the generation options and the valid environments for each option.

The default is `/PREPFIL`.

`/NOPREPFIL` is ignored if the `/PREP` option is specified.

`/PRINTDEST` (Print destination)

Specifies the destination of your printed output for batch programs generated for CICS environments.

This option is only used to set the initial value of the `EZEDESTP` special function word. If a value has already been set for the `EZEDESTP` special

function word, this option is ignored. If a value is not set for EZEDESTP, then the value specified for /PRINTDEST is used to determine the print destination.

You can specify the following values:

EZEPRINT

Select **EZEPRINT** if you want the printed output from CICS batch jobs sent to the destination specified for the EZEPRINT file.

TERMID

Select **Terminal ID** if you want the printed output from CICS batch jobs sent to the current CICS terminal ID.

This is compatible with the behavior of the Cross System Product set.

The default is EZEPRINT.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/PROJECTID (Project ID)

Specifies the value that you want to use as the high-level data set name qualifier in the preparation and run-time JCL.

The value you specify is substituted for the template symbolic parameter %EZEPID%. EZEPID is used in JCL and CLIST templates as the high-level data set name qualifier for the data sets that contain the preparation output files. The value of /PROJECTID can be your user logon ID or a project ID. The maximum length of /PROJECTID is 54 characters.

The value can also be used to route generation outputs for different projects or users to different directories by specifying %EZEPID% as one of the subdirectories on the /GENOUT option path.

Table 77 on page 375 shows the generation options and the valid environments for each option.

The default is PROJECT; however, you can specify a value for /PROJECTID if generation is to create host JCL.

/RECOVERY (Recover current error message)

Specifies that you want the message currently being processed inserted in the message queue again if the program ends because of an error.

Specify `/NORECOVERY` if you do not want the current message inserted in the message queue again.

This option applies only if the `SCAN` process option is used to read the message queue in a transaction-oriented IMS BMP program. If a `CSPTDLI` call is used to read the message queue, the message is not inserted in the message queue again.

This option is effective at run time only if the option is specified for the first program in the run unit.

Table 77 on page 375 shows the generation options and the valid environments for each option.

The default is `/RECOVERY` for a transaction-oriented IMS BMP that uses the `SCAN I/O` option to read the message queue, and `/NORECOVERY` for other IMS BMP programs.

/RESOURCE (Resource associations)

Specifies the name of a resource association part containing resource associations.

The contents of a resource association part indicate where the physical files used by the program are located.

Note: The resource association part is used during the generation process for C++ programs only when the target environment is CICS for AIX, CICS for Solaris, CICS for OS/2, or CICS for Windows NT. When the target environment is OS/2, AIX, HP-UX, SCO, Solaris, or Windows NT, C++ programs use resource association files at run time. The contents of the resource association file indicate where the physical files used by the program are located.

Refer to the *VisualAge Generator Server Guide for Workstation Platforms* for information about the resource association file and supported file types for the OS/2, AIX, HP-UX, SCO, Solaris, and Windows NT environments.

Table 77 on page 375 shows the generation options and the valid environments for each option.

There is no default value.

See “Chapter 28. Resource associations part” on page 265 for more information about resource associations.

/RESOURCEBUNDLELOCALE

Specifies a Java locale to be included in the name of a resource bundle generated for a UI record or a Web transaction message table. The option RESOURCEBUNDLELOCALE is used in the following cases:

- You are generating only a UI record or message table and have specified option GENRESOURCEBUNDLE.
- You are generating a Web transaction; in this case, if you have specified option GENRESOURCEBUNDLE, one resource bundle is generated for each UI record and one for each message table; but if you have not specified option GENRESOURCEBUNDLE, a resource bundle is generated only for the message table.

A resource bundle is a Java object that contains strings to be presented at runtime. The content of a resource bundle generated for a UI record includes the web-page title, labels, and help text; and the content of a resource bundle for a message table includes the set of messages. A resource bundle is specific to a human language, and you can make a web transaction clear to a wider audience by translating resource bundles into different languages.

The name of each generated resource bundle is as follows:

`NameRBundle_JavaLocale.java`

Name is the UI record name or message table prefix. If you generate a resource bundle without specifying a resource bundle locale, the generated resource bundle name is as follows:

`NameRBundle.java`

At runtime, as a result of a browser setting, the user who invokes a Web transaction specifies a *Java locale*, which is a code identifying a human language. Each subset of information in the Java locale is separated from the next by an underscore. For example, *en* represents English, *en_US* represents United States English (distinct from British English). You could also add a third value as in *no_NO_B*, which specifies a particular variant of Norwegian. The first two items—the *language code* in lower case and the *country code* in uppercase are based on a Java language specification; and any additional, items constitute a *variant*, which is not part of a Java specification.

If the user requests a Java locale that is precisely matched in the name of an available resource bundle, that resource bundle is a source of strings presented to the user. The user may requests a Java locale that is more specific than the locale in the name of the available resource bundle—for example, the user may specify *no_NO_B* when the only the Norwegian-language resource bundle is named with *no_NO*; in this case, the less-specific bundle is a source.

It is good practice to provide *default resource bundles*, the names of which include no Java locale. A default resource bundle becomes the source if the user specifies either no locale or a locale that is less specific than the locale in the names of other bundles. If the user specifies *en* when the only locale-specific bundle includes the name *en_US*, for example, the default bundle is accessed.

At runtime, if no resource bundle can be loaded for a UI record, the string values specified at definition time are presented to the user. The lack of a resource bundle for a message table results in runtime errors if the generated program invokes a message table.

In relation to web transactions, the generation option `/TARGNLS` identifies the name of the message table from which a resource bundle is generated. The name of the message table is as follows:

PrefixNLSvalue

Prefix is the message table prefix and *NLSvalue* is the setting of `/TARGNLS`.

`/TARGNLS` is no longer used to determine the name of the resource bundle itself during message-table generation. Use of `RESOURCEBUNDLELOCALE` for this purpose allows support of more languages than those for which IBM supplies translation.

There is no default value.

Table 77 on page 375 shows the generation options and the valid environments for each option.

`/RESVWORD` (Reserved words)

Specifies the file that contains the reserved words, such as COBOL, SQL, and CICS reserved words, for your project.

Specify the directory name as a fully qualified directory name. Symbolic parameters are permitted in the directory specification.

See “Chapter 12. Symbolic parameters” on page 123 for more information about symbolic parameters. Partial directory names are treated as a subdirectory of the current server process directory. See “Chapter 15. Command interface for COBOL generation” on page 167 for more information about the server process.

Table 77 on page 375 shows the generation options and the valid environments for each option.

The default file name is `EFK2RSV.RSV`.

See “Reserved-word file” on page 53 for more information about reserved words.

/RT (Return or Return transaction ID)

Specifies a transaction identifier.

The strings for transaction identifiers are set in the following ways, depending on the environments:

CICS and IMS/VS

Specifies the transaction identifier that is started when the transaction associated with the VisualAge Generator program ends without transferring

If you do not specify this option, the default action is to clear the screen when the program ends without transferring to a new transaction. You can prevent the message to clear the screen from being sent in the IMS environment by specifying the value /RT=EZENOINS (no message insert at all) when generating the main program for IMS.

Only the first 4 characters are used for CICS environments; any remaining characters are ignored.

MVS/TSO and VM CMS

Specifies a program name rather than a transaction identifier

In MVS/TSO, data sets must already be allocated for the target program as if they were the target of an XFER statement.

This option is effective at run time only if the option is specified for the first program in the run unit.

You usually use this method of specifying a transaction identifier when a menu-driven program starts the program and you want to return to the menu.

This option applies to the following types of programs:

- Main transaction programs in IMS/VS
- Main transaction and main batch programs in MVS/TSO, VM CMS, and all CICS environments

Table 77 on page 375 shows the generation options and the valid environments for each option.

/RUNFILE (Create sample run-time JCL, Create a sample clist, or Create a sample REXX exec)

Specifies that you want to generate run-time files.

Specify `/NORUNFILE` if you do not want to generate run-time files.

Run-time files consist of the following files:

- Sample run-time JCL for MVS batch, IMS BMP, or VSE batch programs (*prgmname.JCX*)
- A sample CLIST for MVS/TSO programs (*prgmname.CLX*)
- A sample run-time REXX exec for VM CMS and VM batch programs (*prgmname.RXX*)

Table 77 on page 375 shows the generation options and the valid environments for each option.

The default is `/RUNFILE`.

/SENDTRANSLATIONCMDDBCS (Send DBCS Translation Command)

Specifies that the DBCS translation tables must be set for the remote system when transferring files using SNA. Set this value if you need to specify other translation tables on the remote system. This is normally not necessary, because any needed translation is done automatically on the local system.

Table 77 on page 375 shows the generation options and the valid environments for each option.

Refer to your remote system documentation for default values.

/SESSION (Session ID)

Specifies the session ID of the host emulator session you want to use when generated files are transferred to the host.

When this option is used with the `GENERATE` subcommand, the `/PREP` generation option must also be specified.

Table 77 on page 375 shows the generation options and the valid environments for each option.

The default is the first host session ID configured for your system.

/SETFULL (Set map item FULL)

Specifies that you want the VisualAge Generator `SET` map-item `FULL` statement supported by the print services programs.

Specify `/NOSETFULL` if you want the VisualAge Generator SET map-item FULL statement ignored by the print services programs.

Specify `/SETFULL` if you want asterisks (*) displayed in an empty map variable if SET map item FULL statement is performed for the variable.

Specifying `/NOSETFULL` might provide better performance and smaller load module size for map groups containing printer maps with many map variables.

Specify `/NOSETFULL` if you do not use SET map item FULL statement in the programs that use the map group.

Note: This option affects only the generation of map groups that contain printer maps. It has no effect on the generation of programs, tables, or map groups that only contain terminal maps.

Table 77 on page 375 shows the generation options and the valid environments for each option.

The default is `/SETFULL`.

`/SP (Issue CICS SET/INQUIRE)`

Specifies the use of system program commands.

`/SPA (SPA)`

Specifies the size of the IMS scratch-pad area (SPA) you want to use for transaction programs.

Table 77 on page 375 shows the generation options and the valid environments for each option.

The default value is 0.

`/SPZERO (Interpret spaces as zero in NUM and NUMC data items)`

Specifies that you want NUM and NUMC data items that contain only spaces to be interpreted as containing all zeros.

You need not specify `/NOSPZERO` if you do not want special processing done for NUM and NUMC data items that contain only spaces; `/NOSPZERO` is the default.

Specify the `/SPZERO` option to generate COBOL source code that ensures NUM and NUMC data items are interpreted as zeros instead of spaces. Run-time performance is affected.

You do not need to specify the /SPZERO option if the NUM and NUMC data items that contain all spaces cannot be used in an assignment statement or in a conditional expression.

This option applies only to NUM and NUMC data items that contain just spaces; this option does not apply to other types of data items.

Note: You cannot use nonnumeric data, such as national characters and spaces, in NUM or NUMC data types. This option does not interpret data items that contain a combination of spaces and other data.

An abend results if a numeric data type that contains character data is used in an assignment statement or in a conditional expression. A data exception causes this abend.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/SQLDB (SQL database)

Specifies the name of the database you want to use for SQL statement validation.

When you use this option with the VisualAge Generator GENERATE subcommand, you must also specify the /PREP generation option.

The default value for /SQLDB is the value specified for the environment variable EZERSQLDB.

Notes:

- If you specify this option in a command file and you want to use a symbolic parameter as part of the value for this option, you must specify two percent signs (%%) as the symbolic parameter delimiter.
- If you specify this option on the command line or in a generation options file, the symbolic parameter delimiter is one percent sign (%).
- The user interface always places the options in a command file. If you use symbolic parameters in values specified for the interactive interface, always use two percent signs as the symbolic parameter delimiter. For example, to use the symbolic parameter EZEENV, specify %%EZEENV%%. This ensures that the command interpreter correctly replaces the symbolic parameter value.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/SQLID (SQL userid)

Specifies the name of the SQL ID to use for SQL statement validation.

Notes:

- If you specify this option in a command file and you want to use a symbolic parameter as part of the value for this option, you must specify two percent signs (%%) as the symbolic parameter delimiter.
- If you specify this option on the command line or in a generation options file, the symbolic parameter delimiter is one percent sign (%).
- The user interface always places the options in a command file. If you use symbolic parameters in values specified for the interactive interface, always use two percent signs as the symbolic parameter delimiter. For example, to use the symbolic parameter EZEENV, specify %%EZEENV%%. This ensures that the command interpreter correctly replaces the symbolic parameter value.

The default value is the value specified for the environment variable EZERSQLID.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/SQLPASSWORD (password)

Specifies the name of the SQL password to use.

Notes:

- If you specify this option in a command file and you want to use a symbolic parameter as part of the value for this option, you must specify two percent signs (%%) as the symbolic parameter delimiter.
- If you specify this option on the command line or in a generation options file, the symbolic parameter delimiter is one percent sign (%).
- The user interface always places the options in a command file. If you use symbolic parameters in values specified for the interactive interface, always use two percent signs as the symbolic parameter delimiter. For example, to use the symbolic parameter EZEENV, specify %%EZEENV%%. This ensures that the command interpreter correctly replaces the symbolic parameter value.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/SQLVALID (SQL statements)

Specifies that you want to validate SQL statements.

You need not specify `/NOSQLVALID` if you do not want to validate SQL statements; `/NOSQLVALID` is the default.

SQL statements are validated by dynamically preparing the statement using the current database manager on the generation system. Validation of SQL statements is done according to current database manager dynamic SQL preparation rules.

For validation against remote databases, ensure that the database is catalogued in the DDCS directory.

SQL statement validation does not catch all errors. The validation process might return errors for the SQL statements that are valid in the target environment, but not valid for the current database manager.

Note: You cannot validate SQL statements for SQL records defined with dynamic table names or statements with the Execution Time Statement Build option specified as Yes.

To validate SQL statements, all databases and tables used in the statements must already be defined to the current database manager.

Requesting SQL statement validation increases the time needed for validation.

The default is `/NOSQLVALID`.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/SYMPARM

Assigns a value to a symbolic parameter

The VisualAge Generator Developer builds JCL, CLIST, preparation command files, the JOB and CICS program and transaction definitions, and preparation and run-time REXX exec files from template files. If a symbol name in a template matches a symbol you specified with the `/SYMPARM` option, the value specified for the option replaces the name in the template.

Multiple `/SYMPARM` options can be specified on the same `GENERATE` subcommand.

Note: The `/SYMPARM` generation option can only be specified using the `GENERATE` subcommand or a generation options part. The user interface has no corresponding field for the `/SYMPARM` generation option.

Symbols specified using the /SYMPARM option cannot begin with EZE. Symbols can be 1- to 8-characters in length.

Values specified for symbolic parameters must be enclosed in single quotation marks (') or double quotation marks (").

Table 77 on page 375 shows the generation options and the valid environments for each option.

See "Chapter 12. Symbolic parameters" on page 123 for more information about symbolic parameters.

/SYNCDXFR (Set sync points for DXFRs)

Specifies that you want synchronization points to occur on DXFR operations whenever a PSB is scheduled.

Specify /NOSYNCDXFR if you do not want synchronization points to occur on DXFR operations unless a PSB is scheduled and the transferred-to program was defined with a different PSB than the transferring program. The PSB name in the program specification for both programs must be the same to avoid the synchronization point at the DXFR operation.

Specifying /NOSYNCDXFR makes processing in CICS environments compatible with other environments.

The default is /SYNCDXFR.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/SYNCXFER (Set sync points for XFERs)

Specifies that you want to force database commits on XFER operations in MVS/TSO, VM CMS, IMS BMP, MVS batch, and VM batch systems. Specifying /SYNCXFER forces compatibility across all environments.

Specifying /NOSYNCXFER indicates that commits are not performed on XFER operations in these environments.

/SYNCXFER is not supported in transaction-oriented IMS BMP programs (programs which SCAN the I/O PCB as a serial file). These programs automatically commit database changes on each SCAN to the I/O PCB and cannot commit changes at any other point.

The default is /NOSYNCXFER.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/SYS_CODES (Use system return codes)

Specifies that you want the system return codes returned in your program.

You need not specify /NOSYS_CODES to indicate that you want your program to return VisualAge Generator return codes in the special function word EZERT8 following I/O options; /NOSYS_CODES is the default.

The default is /NOSYS_CODES.

Table 77 on page 375 shows the generation options and the valid environments for each option.

For more information about system return codes for the format of VisualAge Generator return codes, refer to the *VisualAge Generator Messages and Problem Determination Guide*.

/TARGNLS (Target NLS)

Specifies the target national language code used for run-time output.

TARGNLS is the target national language code.

In relation to web-transaction programs, the following is true:

- TARGNLS identifies the name of the message table from which a resource bundle is generated. The name of the message table is as follows:

PrefixNLSvalue

Prefix is the message table prefix and *NLSvalue* is the setting of TARGNLS.

TARGNLS is no longer used to determine the name of the resource bundle itself during message-table generation. Use of RESOURCEBUNDLELOCALE for this purpose allows support of more languages than those for which IBM supplies translation.

- TARGNLS also specifies the default code-point-conversion table that is available when the web-transaction program itself acts as a client and either uses a linkage table or includes EZECONV or EZECONVT.

The rest of the information provided here concerns use of TARGNLS outside of web-transaction programs.

TARGNLS defines the following specifications:

- The language for user messages and host services or workgroup services messages at run time

- The language for run-time messages that are generated into a map group generated for message format service (MFS)
- The fold table used at run time
- The default code point conversion table used in client/server programs at run time
- The default code point conversion table used for generating map group format modules for MVS, VSE, and VM environments
- The default conversion table used for translating ASCII to EBCDIC when transferring objects from the workstation to MVS, VSE, and VM systems for preparation

The /TARGNLS option is a run unit option for programs.

The /TARGNLS code for the first program in the run unit determines the language you want to use for error messages for all programs in the run unit.

The /TARGNLS option acts as an option at generation (when generating messages into the MFS source and identifying the default conversion tables).

The national language generated with the map group cannot be changed at run time. To ensure that the national language values used in a program and its map groups are consistent, specify the same /TARGNLS code when generating the program and its map groups.

The following languages are supported:

Code	Language
CHS	Simplified Chinese
CHT	Traditional Chinese
DES	Swiss German
DEU	German
ENP	Uppercase English
ENU	US English
ESP	Spanish
FRA	French
ITA	Italian
JPN	Japanese
KOR	Korean
PTB	Brazilian Portuguese

Note: Uppercase English is not supported by AIX, OS/2, Windows NT, HP-UX, SCO OpenServer, and Solaris.

Note: You must have the code pages loaded on your system for the languages you use as values for the target national language.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/TEMPLATES (Templates directory)

Specifies the directory that is searched for templates used in generating JCL, CLISTS, REXX execs, preparation command files, the JOB statement, and CICS program and transaction definitions.

If the templates are not found, the directories specified by the DPATH environment variable are searched. Use the /TEMPLATES option if you built different versions of the templates for different projects. Specify the directory name as a fully qualified directory name. Symbolic parameters are permitted in the directory specification.

See “Chapter 12. Symbolic parameters” on page 123 for more information about symbolic parameters. Partial directory names are treated as a subdirectory of the current server process directory. See “Chapter 15. Command interface for COBOL generation” on page 167 for more information about the server process.

Notes:

- If you specify this option in a command file and you want to use a symbolic parameter as part of the value for this option, you must specify two percent signs (%%) as the symbolic parameter delimiter.
- If you specify this option on the command line or in a generation options file, the symbolic parameter delimiter is one percent sign (%).
- The user interface always places the options in a command file. If you use symbolic parameters in values specified for the interactive interface, always use two percent signs as the symbolic parameter delimiter. For example, to use the symbolic parameter EZEENV, specify %%EZEENV%%. This ensures that the command interpreter correctly replaces the symbolic parameter value.

Table 77 on page 375 shows the generation options and the valid environments for each option.

See “Chapter 11. Templates for COBOL generation” on page 57 for more information about templates.

/TRACE (Runtime trace)

Specifies that you want to generate the necessary COBOL statements for run-time tracing to be performed.

You can specify the following values:

NONE

Specify **NONE** if you do not want SQL tracing performed at run time.

Note: Because setting the /TRACE option to a value other than **NONE** generates additional COBOL statements, using this option will affect the run-time performance of the generated program even if tracing is not active.

SQLERR

Specify **SQLERR** to generate the COBOL statements needed to trace SQL error codes from the SQL communications area (SQLCA) at run time.

SQLIO

Specify **SQLIO** to generate the COBOL statements needed to trace SQL data areas and SQL error codes from the SQLCA at run time.

STMT

Specify **STMT** to enable VisualAge Generator Developer statement tracing at run time. Statements are traced only if you specify **STMT** when you run the program.

Note: If SQLERR or SQLIO is specified for a program that does not contain SQL statements, the specification of SQLERR or SQLIO is ignored.

For more information about enabling tracing and for information about obtaining the trace output, refer to any of the following documents:

- *VisualAge Generator Server Guide for MVS, VSE, and VM*
- *VisualAge Generator Server Guide for AS/400*
- *VisualAge Generator Server Guide for Workstation Platforms*

Table 77 on page 375 shows the generation options and the valid environments for each option.

The default is NONE.

/TRANSFERTYPE (Transfer method)

Specifies the protocol used for transferring files to the remote system.

The default values are as follows:

Development System	CICS for OS/2	AS/400	MVS	VM	VSE
Windows NT	TCPIP	TCPIP	SNA	SNA	SNA
OS/2	NONE	TCPIP	SNA	SNA	SNA

If you are using the SNA protocol, you can specify /SESSION. If you are using the TCP/IP protocol, you must specify /DESTHOST, /DESTOPASSWORD, and /DESTUID.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/TRANSID (Transaction IDs)

In relation to a main transaction or a Web transaction, you can specify the following values for use in CICS environments:

primary

The trans-id that starts the program. The default is the first four characters of the program name.

The specification of primary value has no effect at run time, but causes the generation of PCT and RDO entries in accordance with the setting of generation option /CICSENTRIES.

For a Web transaction, you must have PCT entries for the catcher program DFHMIRS and for each Web transaction that DFHMIRS can invoke. The /TRANSID primary value identifies a set of PCT and RDO entries to be generated.

restart The trans-id that restarts the program. The restart occurs when the user submits data in response to a segmented CONVERSE option.

At runtime, the restart value initializes the EZE word EZESEGTR. The default for restart is a blank, which causes VisualAge Generator to initialize EZESEGTR with the trans-id that the user entered to invoke the program.

Like the primary value, the restart value causes the generation of PCT and RDO entries in accordance with the value of generation option /CICSENTRIES.

In a Web transaction, the restart value (if any) refers to the trans-id for the catcher program DFHMIRS. To avoid starting CPMI when the trans-id is not CPMI, prepend tpn_ to the trans_id. If the trans-id is WEBT, for example, specify tpn_WEBT; but if the trans-id is CPMI, specify only CPMI. If you specify WEBT without tpn_, CICS starts CPMI, which in turn switches control to WEBT.

For details on EZESEGTR, see the *VisualAge Generator Design Guide*. For details on Web transactions, see the *Web Transaction Development Guide*.

Table 77 on page 375 shows the generation options and the valid environments for each option.

/TWAOFF (TWA offset)

Specifies the offset of the 1024-byte block in the TWA. By default, generated programs use 1024 contiguous bytes in the CICS transaction work area (TWA) to keep track of transaction status

This option is effective at run time only if the option is specified for the first program in the run unit.

Use this option to avoid overlap when non-VisualAge Generator programs that use the transaction work area (TWA) are in the same run unit with COBOL programs.

For example, if your non-VisualAge Generator program requires 100 bytes of TWA, specifying a value of 100 causes the first 100 bytes to be reserved for your program.

For CICS for OS/2 generation, when you specify a nonzero value you must increase the TWASIZE defined in the transaction definition for the System Initialization Table (SIT) by the amount beyond the 1024 bytes required for all VisualAge Generator transactions.

For CICS for MVS/ESA and CICS for VSE/ESA, the increase in TWASIZE is automatically generated in the transaction definitions. The TWASIZE value generated in the PCT is set to 1024 plus the value specified for the /TWAOFF generation option.

You can specify the following values:

0

Specify 0 if you want generated programs to use the first 1024 bytes of the TWA.

offset

Specify offset to specify the offset of the 1024-byte transaction status area in the TWA used by generated programs.

Table 77 on page 375 shows the generation options and the valid environments for each option.

The default is 0.

/UNLOAD (Unload parts)

Specifies that all applications (on Smalltalk) or projects (on Java) containing VAGen parts are to be unloaded before a batch generation.

The sequence is as follows:

1. An unload of all VAGen parts occurs.
2. The configuration map or project specified on the Generate command is loaded.
3. Generation occurs.

Table 77 on page 375 shows the generation options and the valid environments for each option.

The default is /UNLOAD.

/VALIDMIX (Validate mixed field moves)

Specifies that mixed fields are validated by VisualAge Generator Server or VisualAge Generator Server for MVS, VSE, and VM.

Mixed fields are fields that contain both double-byte character set (DBCS) and single-byte character set (SBCS) characters. Specify /NOVALIDMIX if you want validation of mixed fields handled by the COBOL program. Using /NOVALIDMIX might result in improved performance depending on the number of mixed field moves contained in the program.

Specify the /VALIDMIX option to ensure that DBCS strings are valid, especially if mixed fields might be truncated.

You do not need to specify the /VALIDMIX option if there is no possibility of truncation of mixed fields.

Specify the /NOVALIDMIX option if you do not require the VisualAge Generator Server to validate double-byte strings.

Table 77 on page 375 shows the generation options and the valid environments for each option.

The default is /VALIDMIX.

/VMLOADLIB (VM Load Library)

Specifies the VM target load library where the load modules are placed by the preparation process.

Load modules created during the preparation process are stored in this library.

The value for /VMLOADLIB is the file name for the load library; the filetype is always LOADLIB.

The file name is limited to 8 characters.

Notes:

- If you specify this option in a command file and you want to use a symbolic parameter as part of the value for this option, you must specify two percent signs (%%) as the symbolic parameter delimiter.
- If you specify this option on the command line or in a generation options file, the symbolic parameter delimiter is one percent sign (%).
- The user interface always places the options in a command file. If you use symbolic parameters in values specified for the interactive interface, always use two percent signs as the symbolic parameter delimiter. For example, to use the symbolic parameter EZEENV, specify %%EZEENV%%. This ensures that the command interpreter correctly replaces the symbolic parameter value.

Table 77 on page 375 shows the generation options and the valid environments for each option.

The default value for /VMLOADLIB is EZEAPPL.

/LIB (VSE Library)

Specifies the library where objects created during generation are placed when transferred by the preparation process.

Objects used in preparation, such as source code and text decks from COBOL compiles, are placed in this library.

The parts are specified using the syntax *part.type*. This library is the first library searched for link-editing.

The library specification is limited to 16 characters.

Notes:

- If you specify this option in a command file and you want to use a symbolic parameter as part of the value for this option, you must specify two percent signs (%%) as the symbolic parameter delimiter.
- If you specify this option on the command line or in a generation options file, the symbolic parameter delimiter is one percent sign (%).
- The user interface always places the options in a command file. If you use symbolic parameters in values specified for the interactive interface, always use two percent signs as the symbolic parameter delimiter. For example, to use the symbolic parameter EZEENV, specify %%EZEENV%%. This ensures that the command interpreter correctly replaces the symbolic parameter value.

Table 77 on page 375 shows the generation options and the valid environments for each option.

The default value for /LIB is PRD2.EZELIB.

/WORKDB (Work database)

Specifies the type of work database that you want to use for the generated program.

This option is effective at run time only if the option is specified for the first program in the run unit.

The work database is used for the following purposes:

- Saving program status across a CONVERSE process option in a segmented program in IMS/VS and all CICS environments
- Saving input from a map while an end user views a help map
- Saving output for a map if the EZEMSG field was not included on the map and an error message needs to be displayed in IMS/VS and all CICS environments
- Saving a copy of the map for an XFER statement with a map in IMS/VS and all CICS environments
- Passing working storage on an XFER statement in the IMS/VS environment

IMS environments support the definition of multiple DL/I databases, or SQL databases, or both. Therefore, both the transferring and transferred-to package/application must use not only the same /WORKDB type but also the same physical database.

You can select any of the following values:

DLI

Specify DLI if you want to use DLI as the work database.

DLI is a valid value for the IMS/VS environment.

If you specify DLI, then you must include the ELAWORK PCB in the program PSB.

The default for IMS/VS is DLI if you specified the ELAWORK database in the program PSB.

SQL

Specify SQL if you want to use SQL as the work database.

SQL is a valid value for the IMS/VS environment.

The default is SQL if the ELAWORK database is not defined in the program PSB. If there is an ELAWORK PCB in the PSB, DLI is the default. SQL can be used in this case to override the DLI value.

AUX

Specify AUX if you want to save program status data in an auxiliary temporary storage queue.

AUX is a valid value for all CICS environments.

The default is AUX for CICS environments.

MAIN

Specify MAIN if you want to save program status data in a main temporary storage queue.

MAIN is a valid value for all CICS environments.

You can specify MAIN for CICS for OS/2 environments; however, the current release of CICS for OS/2 always puts temporary storage queues on auxiliary storage, even when the generated program specifies MAIN.

Table 77 on page 375 shows the generation options and the valid environments for each option.

Refer to the *VisualAge Generator Design Guide* for a description of work database considerations for IMS.

Chapter 30. Java properties files

Java properties files contain settings that are used by Java programs at runtime. You can elect to have a properties file generated when you generate a program or you can write your own. From the Generation Options notebook, you can specify generation of a properties file. If you are doing batch generation, you specify generation of a properties file using `/GENPROPERTIES`.

You can also modify a properties file after you have generated it to change the way your program runs without regenerating it. This chapter provides a list of properties you can use in the properties file with an explanation of what options or preferences they correspond to. The properties are arranged by the purpose they serve at runtime.

Resource association properties

Properties that correspond to resource association specifications are shown in the following list. In the examples shown here, `<filename>` is taken from "ASSOCIATE FILE=`filename`" specification in the resource association file.

`vgj.ra.<filename>.filetype`

Specifies the type of the associated file.

This property is defined in the resource association file by the `/FILETYPE` option. It is a required property. The following values are valid:

- `SEQ` specifies a sequential file
- `MQ` specifies a message queue.

In the properties file, a sequential file type specification is shown as:

```
vgj.ra.MYFILE.filetype=SEQ
```

`vgj.ra.<filename>.replace`

Specifies whether or not an existing serial file is to be replaced by a new one.

This property is defined in the resource association file by the `/REPLACE` option. The following values are valid for this property:

- A value of 1 specifies that a file is to be replaced.
- A value of 0 specifies that the file is not to be replaced and is the default.

You can use a value of 0 to specify that an `ADD` to a file always appends to the end of the file.

In the properties file, a replacement specification would be shown as:
`vgj.ra.MYFILE.replace=1`

vgj.ra.<filename>.sysname

Specifies the system name of the file associated with the file name in the record.

This property is defined in the resource association file by the /SYSNAME option. This is a required property. The format of the name is system and type-dependent as shown in the following examples:

- If the file type is SEQ, `vgj.ra.<filename>.sysname` specifies the fully qualified path and file name. (Slashes in path names are special characters that must be escaped using a back slash.) In the properties file, a system name for a sequential file would be shown as:

`vgj.ra.MYFILE.sysname=c:\\docs\\customers.txt`

- If the file type is MQ, `vgj.ra.<filename>.sysname` specifies the queue manager name and the queue name. In the properties file, a system name specification for a message queue would be shown as:

`vgj.ra.MYFILE.sysname=queue_manager_name:queue_name`

vgj.ra.<filename>.text

Specifies a special type of serial file that contains carriage return or line feed (CRLF) characters at the end of each line.

When a record is read (SCAN), the CRLF characters are automatically removed. When a record is written (ADD), the CRLF characters are automatically appended to the end of each record. This option is useful when exchanging data files with other software packages that expect each record to be delimited with the CRLF characters. This property is defined in the resource association file by the /TEXT option. The following values are valid for this property:

- A value of 1 specifies that the file contains CRLFs.
- A value of 0 specifies that the file should be read as bytes. 0 is the default.

In the properties file, a text file specification would be shown as:

`vgj.ra.MYFILE.text=1`

vgj.ra.<filename>.contable

Specifies the conversion table used when the record data comes from a message queue. The following values are valid for this property:

- * specifies use of the default conversion table.
- EZECONVT specifies that the conversion table name it contains should be used as the conversion table.

- `<conversiontablename>` specifies the name of the conversion table.
- `NONE` specifies that no conversion table is used. This property is the default.

In the properties file, a conversion table specification would be shown as:

```
vgj.ra.MYFILE.contable=EZECONVT
```

Database default properties

Java SQL programs can use these properties for default access to SQL database.

vgj.jdbc.default.database

Specifies the default database name to be used for an SQL I/O operation if no prior database connection is specified. In a generated properties file, this value comes from the SQL database (`/SQLDB`) generation option.

In the properties file, a default database is specified as follows:

```
vgj.jdbc.default.database=emplsdb
```

vgj.jdbc.default.database.user.id

Specifies the default database user ID to be used with the default database connection if a default connection is specified. In a generated properties file, this value comes from the SQL userid (`/SQLID`) generation option.

In the properties file, a default database user ID is specified as follows:

```
vgj.jdbc.default.database.user.id=vguser
```

vgj.jdbc.default.database.user.password

Specifies the default database user password to be used with the default database connection if a default connection is specified. In a generated properties file, this value comes from password (`/SQLPASSWORD`) generation option. Use `EZECONCT` to avoid exposing passwords in the properties file.

In the properties file, a default database password is specified as follows:

```
vgj.jdbc.default.database.user.id=vgpasswd
```

vgj.jdbc.drivers

Specifies the names of one or more JDBC drivers to be used for JDBC access. If multiple drivers are specified, the names should be separated by colons.

In the properties file, a JDBC driver is specified as follows:

```
vgj.jdbc.drivers=COM.ibm.db2.jdbc.app.DB2Driver
```

vgj.jdbc.database.<servername>

Specifies the name of the JDBC database that is used with the default database specification when a call to EZECONCT is made with the specified server name.

In the properties file, a server name is specified as follows:

```
vgj.jdbc.database.DB2SERV=jdbc.db2.sample
```

JVM command property

Although the default command for starting a JVM is `java`, in the properties file you can define a different command to start a new JVM. You should use this property when you start a program using a CREATX statement.

vgj.java.command

Specifies an alternate command to be used at runtime.

In the properties file, an alternate start command is specified as follows:

```
vgj.java.command=jre
```

You can also use this property to specify both a different start command and a different properties file:

```
vgj.java.command=jre -Dvgj.properties.file=c:\\java\\alternatename.properties
```

Note: For additional information about using the `vgj.properties.file` property, see the *VisualAge Generator Server Guide for Workstation Platforms*.

Java server communication properties

If the program you are setting up is a Java server program that uses TCP/IP you will need to start the following service:

CSOTcpipListener

A Java program that handles TCP/IP calls. You start this program using the command:

```
java CSOTcpipListener
```

If you do not specify a path to the properties file for the TCP/IP listener, the Java virtual machine (JVM) looks for the default properties file `tcpiplistener.properties` in your current directory.

If the program you are setting up or developing is a Web transaction, running your program requires that the TCP/IP listener service be running. The following service must also be running:

CSOUIListener

A Java program used to start Web transactions. You start this program using the command:

```
java CSOUIListener
```

Each of these services requires a properties file. You can specify the fully qualified path to these properties files on the command line as follows:

```
java CSOUIListener c:\java\csoul.properties
```

If you do not specify a path to the properties file for the UI listener, the Java virtual machine (JVM) looks for the default properties file `uilistener.properties` in your current directory.

Properties files for both services are structured the same way and use the same properties. In the following summary, `<listener>` can be either `uilistener` or `tcpiplistener`.

<listener>.port

Specifies the number of the port on which the program will listen for connections.

In the properties file, the port number is specified as follows:

```
tcpiplistener.port=9876
```

<listener>.java.command

Specifies whether server programs started by the listener are to run in the same JVM as the listener. If the value of this property is `NONE` or if the property is not defined, all of the server programs run in the same JVM with separate threads. If a command is specified as the value of this property, each server runs in a separate JVM, as shown in the following example:

```
tcpiplistener.java.command=java
```

<listener>.trace.flag

Specifies that operations of the listener should be written to a file. The default value of this property is `0`, so no trace file is created. If you specify any other value, the trace is written to a file called `<listener>.out`. To specify a different name, see *<listener>.trace.file*.

In the properties file, the writing of a trace file is specified as follows:

```
tcpiplistener.trace.flag=1
```

<listener>.trace.file

Specifies that operations of the listener should be written to a file with a specific name. The default value of this property is `<listener>.out`.

In the properties file, the file name and location to which the trace should be written is specified as follows:

NLS Properties

Properties from options affecting which language version of VisualAge Generator Server for Windows NT and the program message tables are used can also be specified in the properties files. When a program is generated the NLS code is picked up from generation options. The date format is picked up from user preferences

vgj.nls.code

Specifies the language to be used at runtime. In a generated properties file, this value comes from the Target NLS (/TARGNLS) generation option.

In the properties file, the target national language property is specified as follows:

```
vgj.nls.code=ENU
```

vgj.datemask.<format>.<length>.<NLS>

Specifies the format and language for dates used at runtime. Here, length and format are determined by user-specified settings in options or preferences. NLS is the code defined by the Target NLS (/TARGNLS) generation option.

In the properties file, an example of a date format and language property specification is:

```
vgj.datemask.gregorian.long.ENU=MM/DD/YYYY
```

Note: Gregorian and julian long formats are supported.

vgj.nls.number.decimal

Specifies the character to be used to designate the decimal point in numbers.

In the properties file, an example of a decimal property specification is:

```
vgj.nls.number.decimal=,
```

Linkage properties

Linkage properties are optional controls that define relationships between programs. These properties, taken from generation options and linkage table parts, specify the type of linkage to be used for calls from one program to another. This list summarizes the properties and gives examples of their uses.

cso.linkagetable.<linktable>

Specifies the name of the linkage table part. This property is used to support run-time binding of linkage tables. The value assigned to the

property is the name of the properties file that contains the table's linkage information. In a generated properties file, this value comes from the Linkage table (/LINKAGE) generation option.

In the properties file, the properties file that contains linkage information is specified as follows:

```
cso.linkagetable.LINKTABLE=linkage1.properties
```

cso.application.<applname>

Specifies that programs with names that match <applname> belong to the same server group. If <applname> ends in an asterisk (wild card), it may be used to specify several programs that use the same naming convention. The value of the property is the name of the server group. In a generated properties file, this value comes from the applname attribute as specified in the linkage table.

In the properties file, the programs using the same naming convention can be designated as a server group using as follows:

```
cso.application.J*=JAVASERVERS
```

cso.serverLinkage.<group>.<attribute>

Specifies a server group and a named linkage attribute (package, bitmode, remotecomtype, etc.) to apply to it. The value is the attribute setting. In a generated properties file, this value comes from the attribute setting in the linkage table.

In the properties file, the programs belonging to a named server group can have the value of an attribute applied to them at runtime as follows:

```
serverLinkage.SERVER.remotecomtype=TCPIP
```

Chapter 31. Analyzing return codes and errors

Errors can occur during program generation. Using the following information, you can determine problems and then continue with your work.

Analyzing generation errors

When you use VisualAge Generator, return codes and messages are issued to indicate whether generation was successful.

Analyzing return codes

The return codes for the GENERATE subcommand are as follows:

- 0 Generation was successful.
- 4 Generation was successful, but validation information messages were issued.
- 8 Generation was not successful.
- 12 The command syntax is not valid.
- 16 The NLS identifier is not valid.

Locating generation error messages

Generation messages are written to the STDOUT destination for OS/2 commands. The default STDOUT destination is the current OS/2 session.

To route the messages to a file instead of displaying them in a window, add the following statement to the end of the GENERATE subcommand.

```
> filename 2>&1;
```

Where *filename* is the name of the file you want the generation messages written to. This is the standard OS/2 technique for routing command messages to a file.

Analyzing messages

If the generation was not successful, do the following steps:

- Review the messages to determine why the generation was not successful.
- Correct the problem.
- Generate the part again.

If you receive a stack overflow condition, check your program for any of the following conditions:

- A large number of statements or conditions on either the WHILE or IF processing statements
- A large number of nested expressions
- A large number of operands

If any of these conditions exists, split the statements in your program. If the problem continues, follow this procedure:

1. Record the available information.
2. Record the situation in which this problem occurs.
3. Save the program that you are working with.
4. Use your electronic link with IBM Service (for example, IBMLink) if one is available, or contact the IBM Support Center. If you contact the IBM Support Center to report an error in VisualAge Generator Developer, have your customer number ready. The program number for VisualAge Generator Developer is .

Refer to the *VisualAge Generator Messages and Problem Determination Guide* for more information about the messages.

You can use the /LISTING generation option to generate a listing of the part source code. See “/LISTING /LISTINGONERROR, /NOLISTING (Generation listing)” on page 326 for more information on using the /LISTING generation option.

Analyzing preparation errors

The HPTCMD PREPARE subcommand issues return codes and messages to indicate if preparation was successful.

Analyzing return codes

The return codes for preparation are as follows:

- | | |
|----|----------------------------------------------------------------------------|
| 0 | The preparation process was successful. |
| 4 | The preparation was successful, but some information messages were issued. |
| 8 | The preparation was not successful. |
| 12 | The command syntax is not valid. |
| 16 | The NLS identifier is not valid. |

Note: The values returned by the PREPARE subcommand only indicate the status of the preparation process itself. The values do not indicate whether any jobs submitted by the PREPARE subcommand are successful.

Locating preparation error messages

If preparation was automatically started by specifying the /PREP option, preparation messages go to the same destination as messages returned by the GENERATE subcommand.

If you use the PREPARE subcommand, messages are written to the STDOUT destination for OS/2 commands. The default STDOUT destination is the current OS/2 session.

To route the messages to a file instead of displaying them in the current OS/2 session, add the following to the end of the PREPARE subcommand:

```
> filename 2>&1;
```

where *filename* is the file to write the preparation messages to. This is the standard OS/2 technique for routing command messages to a file.

Analyzing messages

If the preparation was not successful, do the following:

1. Review the messages to determine the cause of the failure.
2. Correct the problem.
3. If the messages indicate a problem in the generated parts, run the generation process again. You can automatically start the preparation process from generation by using the /PREP option.
4. If the problem was in the preparation process, you can start the preparation process again using the PREPARE subcommand.

Refer to the *VisualAge Generator Messages and Problem Determination Guide* for more information about the messages returned by the PREPARE subcommand.

Chapter 32. Serviceability

If you need to contact the IBM Support Center for assistance with a problem, you might be asked to provide any of the following information about your program:

- The program source that was generated with the `/COMMENTLEVEL` option
See “`/COMMENTLEVEL (Comment level) (COBOL)`” on page 304 for information about the `/COMMENTLEVEL` option.
- The external source format export files for the VisualAge Generator parts that were used to generate the program
- You might be asked to use other generation options:
 - `/DEBUGTRACE`
 - `/TRACE`
 - `/LINEINFO`

For more information about enabling tracing and for information about obtaining the trace output when you are generating C++ programs, refer to the *VisualAge Generator Messages and Problem Determination Guide*.

Part 8. Appendixes

Appendix A. List of valid generation options for each environment

Table 77 shows the generation options and the environments for which each option is valid. An equal sign signifies a value is specified with an option; the rest of the options do not expect a value to be added. The options are in alphabetical order.

Table 77. Generation options

Generation options	COBOL										C++								Java	GUI						
	VM CMS	VM batch	MVS CICS	MVS/TSO	MVS batch	IMS/VS	IMS BMP	VSE CICS	VSE batch	OS/2 CICS	OS/400	OS/2	AIX	AIX CICS	HP-UX	SCO	Solaris	Solaris CICS	Windows NT	Windows NT CICS	Java	Java wrapper	OS/2	Windows	Java	
/ANSISQL	x	x	x	x	x	x	x	x	x		x															
/BIND=			x	x	x	x	x																			
/CHECKTYPE=	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			x	x	x	
/CICSDBCS			x					x																		
/CICSENTRIES=			x					x		x				x						x						
/COMMENTLEVEL=	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x						
/CONTABLE=	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			x	x	x	
/CREATEDDS											x															
/CURRENCY=												x	x	x	x	x	x	x	x	x						
/DATA=	x	x	x	x	x	x	x	x	x																	
/DBMS=												x	x		x				x			x				
/DBPASSWORD=										x		x	x	x	x	x	x	x	x	x						
/DBUSER=												x	x	x	x	x	x	x	x	x						
/DEBUGTRACE	x	x	x	x	x	x	x	x	x	x																
/DESTACCOUNT=	x	x	x	x	x	x	x	x	x	x	x															
/DESTDIR=										x		x	x	x	x	x	x	x	x	x						
/DESTHOST=	x	x	x	x	x	x	x			x	x	x	x	x	x	x	x	x	x	x						

Table 77. Generation options (continued)

Generation options	COBOL														C++								Java	GUI		
	VM CMS	VM batch	MVS CICS	MVS/TSO	MVS batch	IMS/VS	IMS BMP	VSE CICS	VSE batch	OS/2 CICS	OS/400	OS/2	AIX	AIX CICS	HP-UX	SCO	Solaris	Solaris CICS	Windows NT	Windows NT CICS	Java	Java wrapper	OS/2	Windows	Java	
/DESTLIB=											x															
/DESTPASSWORD=	x	x	x	x	x	x	x			x	x	x	x	x	x	x	x	x	x	x	x	x				
/DESTUID=	x	x	x	x	x	x	x			x	x	x	x	x	x	x	x	x	x	x	x	x				
/DXFRCANCEL	x	x	x	x	x	x	x	x	x																	
/DXFRXCTL	x	x	x	x	x		x	x																		
/EJBGROUP=																							x			
/ENDCOMMAREA			x					x		x				x				x		x						
/ERRDEST=						x	x																			
/FASTPATH						x																				
/FOLD	x	x	x	x	x	x	x	x	x	x	x															
/FTPTRANSLATIONCMD DBCS	x	x	x	x	x	x	x	x	x	x	x															
/FTPTRANSLATIONCMD SBCS	x	x	x	x	x	x	x	x	x	x	x															
/GENAUTHORTIMEVALUES																						x				
/GENHELPMAPS	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					
/GENMAPS	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					
/GENOUT=	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
/GENPROPERTIES																							x			
/GENRESOURCEBUNDLE																							x			
/GENRET			x																							
/GENTABLES	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		x	x	x
/GENUIRECORDS																							x			
/GROUPNAME														x				x		x						
/INEDIT=	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					
/INITADDWS	x	x	x	x	x	x	x	x	x	x	x															

Table 77. Generation options (continued)

Generation options	COBOL										C++								Java	GUI						
	VM CMS	VM batch	MVS CICS	MVS/TSO	MVS batch	IMS/VS	IMS BMP	VSE CICS	VSE batch	OS/2 CICS	OS/400	OS/2	AIX	AIX CICS	HP-UX	SCO	Solaris	Solaris CICS	Windows NT	Windows NT CICS	Java	Java wrapper	OS/2	Windows	Java	
/INITRECD	x	x	x	x	x	x	x	x	x	x	x															
/JVADESTDIR=																						x				
/JVADESTHOST=																						x				
/JVADESTPASSWORD=																						x				
/JVADESTUID=																						x				
/JAVASYSTEM=																						x				
/JOB CARD=			x	x	x	x	x	x	x	x	x															
/JOBNAME=			x	x	x	x	x	x	x	x	x															
/JSPRELDIR																					x					
/LEFTJUST	x	x	x	x	x	x	x	x	x	x	x															
/LINEINFO	x	x	x	x	x	x	x	x	x	x	x															
/LINES=	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x				x	x	x
/LINKAGE=	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		x	x	x
/LISTING	x	x	x	x	x	x	x	x	x	x											x			x		
/LOCVALID	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x				x	x	x
/LOG=						x	x																			
/MATH=	x	x	x	x	x	x	x	x	x	x																
/MFSDEV=						x	x																			
/MFSEATTR						x	x																			
/MFSIGNORE						x	x																			
/MFSTEST						x	x																			
/MSGTABLEPREFIX=																					x					
/MSP=	x	x		x	x		x		x																	
/NULLFILL	x	x	x	x	x	x	x	x	x	x																
/NUMOVFL	x	x	x	x	x	x	x	x	x	x											x			x		

Table 77. Generation options (continued)

Generation options	COBOL												C++								Java		GUI			
	VM CMS	VM batch	MVS CICS	MVS/TSO	MVS batch	IMS/VS	IMS BMP	VSE CICS	VSE batch	OS/2 CICS	OS/400	OS/2	AIX	AIX CICS	HP-UX	SCO	Solaris	Solaris CICS	Windows NT	Windows NT CICS	Java	Java wrapper	OS/2	Windows	Java	
/OPTIONS=	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
/PACKAGENAME=																						x	x			
/POSSIGN											x															
/PREP	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					
/PREPFILE	x	x	x	x	x	x	x	x	x	x	x															
/PRINTDEST=			x					x		x				x					x		x					
/PROJECTID=			x	x	x	x	x	x	x	x																
/RECOVERY							x																			
/RESOURCE=	x	x	x	x	x	x	x	x	x	x	x	*	*	x	*				*	x						
/RESOURCEBUNDLELOCALE																						x				
/RESVWORD=	x	x	x	x	x	x	x	x	x	x	x															
/RT=	x		x	x		x		x		x				x					x		x					
/RUNFILE	x	x		x	x		x		x																	
/SENDTRANSLATIONCMD DBCS	x	x	x	x	x	x	x	x	x	x	x															
/SESSION=	x	x	x	x	x	x	x	x	x																	
/SETFULL	x	x	x	x	x	x	x	x	x	x	x															
/SP			x																							
/SPA=						x																				
/SPZERO	x	x	x	x	x	x	x	x	x	x	x															
/SQLDB=	x	x	x	x	x	x	x			x	x	x	x	x	x	x	x	x	x	x	x					
/SQLID=	x	x	x	x	x	x	x			x	x															
/SQLPASSWORD=	x	x	x	x	x	x	x			x	x															
/SQLVALID	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					
/SYMPARM=	x	x	x	x	x	x	x	x	x	x	x										x			x		

Table 77. Generation options (continued)

Generation options	COBOL												C++								Java		GUI				
	VM CMS	VM batch	MVS CICS	MVS/TSO	MVS batch	IMS/VS	IMS BMP	VSE CICS	VSE batch	OS/2 CICS	OS/400	OS/2	AIX	AIX CICS	HP-UX	SCO	Solaris	Solaris CICS	Windows NT	Windows NT CICS	Java	Java wrapper	OS/2	Windows	Java		
/SYNCDXFR			x					x		x																	
/SYNCFER	x	x		x	x		x																				
/SYSCODES	x	x	x	x	x	x	x	x	x	x	x																
/SYSTEM=	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x				
/TARGNLS=	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		x	x	x		
/TEMPLATES=	x	x	x	x	x	x	x	x	x	x	x																
/TRACE=	x	x	x	x	x	x	x	x	x	x																	
/TRANSFERTYPE	x	x	x	x	x	x	x	x	x	x	x																
/TRANSID=			x					x		x				x				x		x							
/TWAOFF=			x					x		x																	
/NOUNLOAD=	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x						
/VALIDMIX	x	x	x	x	x	x	x	x	x	x	x																
/VMLOADLIB=	x	x																									
/LIB=								x	x																		
/WORKDB=			x			x		x		x				x				x		x							
Notes: *Includes Windows 95, and Windows NT.																											
The following characters are used to indicate the level of support:																											
x supported																											
blank not supported																											
* resource association file referenced only at run time																											

Appendix B. Implementing a generation server

Support for generation processing, using VisualAge Generator Developer, can be implemented on a workstation that is shared by multiple programmers, but this requires customized implementation and configuration. Sample REXX programs are provided to help you set up client/server generation. These sample files are placed in the C:\Program Files\Vast directory during installation.

You must customize these files for your specific LAN server and operating system environment to provide the required processing support. These sample programs are command-driven and are not directly available on the user interface.

The user interface for generation can be used to create generation command files that invoke the EFKREQ.CMD or EFKREQ.REX client code if the following is true:

- **CALL EFKREQ** is selected on the **VAGen - Generation** tab of the Preferences notebook.
- **Batch Generation** is selected on the Generate window.

Note: These sample REXX programs are the recommended method for running VisualAge Generator Developer on a generation server. It is recommended that you do not use the NET RUN service.

To run these REXX programs on Windows NT you will need to obtain and install REXX support; this is provided by the product IBM Object REXX for Windows NT and Windows 95 (5801-AAR). You need the interpreter edition (Feature 1569, part number 10J9392).

Table 78 shows the sample files provided for OS/2 and Windows NT.

Table 78. Sample files for OS/2 and Windows NT

Sample file	OS/2	Windows NT
EFKREQ	EFKREQ.CMD	EFKREQ.REX
EFKSERV	EFKSERV.CMD	EFKSERV.REX
EFKLIST	EFKLIST.CMD	EFKLIST.REX

The following sample files are provided:

EFKSERV.CMD, EFKSERV.REX

This REXX file contains commands that receive server requests. The EFKSERV.CMD or EFKSERV.REX program runs on the LAN server and continually checks the server input queue for generation requests. The server input queue is specified in the EFKGEN.INI file. End this server process by pressing **CTRL+C** or by issuing the **EFKREQ STOP** command.

EFKREQ.CMD, EFKREQ.REX

This REXX file contains commands that submit server requests. The EFKREQ.CMD or EFKREQ.REX program runs on the client, gathers generation option information, and places the generate request on the server input queue. EFKREQ also supports command options of **PAUSE**, **STOP**, and **START**. These command options control the processing of EFKSERV.

EFKLIST.CMD, EFKLIST.REX

This REXX file contains commands that list the contents of the server input queue. The EFKLIST program runs on the client and displays a list of all requests waiting to be processed by the EFKSERV program.

EFKGEN.INI

This file defines the locations for different files. This file also defines other parameters used by the REXX command files. This file is installed on the client and server.

LANGEN.TXT

This file contains additional configuration guidance for the implementation of generation clients and generation servers.

Setting up the client

You must do the following when setting up a client:

- For Windows NT, install Object REXX. See “Setting up Object REXX support on Windows NT” on page 384.
- Each client must have a copy of the EFKGEN.INI file. The following modifications must be made before the EFKGEN.INI file can be used:
 - The environment variable `GENERATE_INI` value must be set to the directory containing the EFKGEN.INI file.
 - The *working_dir* parameter must be set to a directory located on the client.
 - The *server_input_dir* and *gen_listing_dir* parameter values must be set to directories located on a shared file system that can be accessed by both generation clients and generation servers. Your VisualAge Generator Developer administrator provides these locations.

Alternatively, if all developer's workstations are identical, they can share a copy of the EFKGEN.INI file on the generation server (see "LAN generation setup" on page 385).

- You must have read/write access to the directory on the server where the input queue is located to submit requests. The location of this directory is defined in the EFKGEN.INI file with the value for the *server_input_dir* parameter.

You can submit server requests even if the EFKSERV program is not running. The EFKREQ program places the generation request in input queue of the server.

The HPTCMD command output listings are placed in the directory specified by the *gen_listing_dir* parameter in the EFKGEN.INI file. The generated source and related program objects are placed on the server machine in the directory specified by the /GENOUT generation option.

When you specify a value for the /GENOUT generation option, use symbolic parameters to avoid conflicts with other generation requests. Figure 86 shows how to use a symbolic parameter.

```
/GENOUT=F:\OUTPUT\%EZEPIID%\%EZEENV%
```

Figure 86. Using symbolic parameters example

If your target system is CICS for OS/2 and your project is USERA, the generated output is placed in the directory F:\OUTPUT\USERA\OS2CICS. If a specified directory does not exist, it is created for you.

Setting up the server

You need to be aware of the following when setting up a server:

- For Windows NT, install Object REXX. See "Setting up Object REXX support on Windows NT" on page 384.
- Each server must have a copy of the EFKGEN.INI file. The following modifications must be made before the EFKGEN.INI file can be used:
 - The environment variable GENERATE_INI value must be set to the directory containing the EFKGEN.INI file.
 - The *working_dir* parameter must be set to a directory located on the server.
 - The *server_input_dir* and *gen_listing_dir* parameter values must be set to directories located on a shared file system that can be accessed by both generation clients and generation servers. Your VisualAge Generator Developer administrator provides these locations.

- The EFKSERV program runs continuously. To stop the EFKSERV program, press **CTRL+C.** or issue the **EFKREQ STOP** command.
- The sample REXX programs use long file names. If your file system does not support long file names you must modify the sample REXX programs to use file names that are between 1- to 8-characters.
- The EFKSERV program writes a log listing the generated programs. This log file is called `Gen_Server.log` and is located in the directory specified for the `gen_listing_dir` parameter in the `EFKGEN.INI` file.

See the `LANGEN.TXT` file for additional information.

Setting up Object REXX support on Windows NT

These tasks are only required on Windows NT. OS/2 provides integrated support for REXX.

- Install Object REXX
- Customize Windows NT so that REXX command files, which have a file extension of `.REX`, are recognized as REXX and invoked under the control of the REXX interpreter. To fully implement this level of REXX support in Windows NT you must:
 1. Check that the Object REXX installation process sets up an association for the `.REX` filetype. You can check the existing associations using this command: `assoc | more` If the association does not exist you can add it from the command line using this command:

```
assoc .rex=REXXscript
```

2. Make sure the Object REXX command processor is somewhere in the `PATH` (or enter it fully qualified in this step) and use **ftype** to link it to the file type, if this has not been done by your command processor setup. You can check the existing **ftype** definitions using this command: `ftype | more`. Ensure that the file type association for `REXXscript` has the following value:

```
REXXscript="c:\ObjRexx\rexx.exe" %1 %* >.REX
```

where `c:\ObjRexx` is the directory where Object REXX is installed. If the **ftype** definition does not exist, or does not match the value shown above, you can create the definition from the command line using this command:

```
ftype REXXscript="c:\ObjRexx\rexx.exe" %1 %* >.REX
```

Note: These two steps are sufficient for running the file with the `EXT` extension and might have been done when your command processor was installed. But, you must modify the **ftype** definition for `REXXscript` to ensure parameters are passed as expected.

3. Finally, set the PATHEXT system environment variable to recognize the EXT extension. This is the key step that allows you to use the command file without extension. To set it permanently, use the environment page of the Windows NT system settings notebook dialog (Control Panel → System → Environment) and create the PATHEXT environment variable (because the default is not shown) with a value of:
`.REX;.COM;.EXE;.BAT;.CMD`

LAN generation setup

The following example lists the tasks needed to setup LAN generation on the server machine.

Setup on the generation server

The following example assumes that VisualAge Generator is installed on a LAN server and that all clients install VisualAge Generator on the same drive and directory and use the same mappings of drives on the file server. You must complete the following tasks on the generation server:

1. Install VisualAge Smalltalk client (directory d:\vast)
2. Install VisualAge Generator Developer
3. For Windows NT, install and tailor Object REXX
4. Be sure the drive supports long names
5. Create the following directories:
 - d:\langen
 - d:\langen\template
 - d:\langen\cmds
 - d:\langen\inputgen
 - d:\langen\outlist
6. Copy the following files from d:\vast to d:\langen\cmds:
 - efkreq.rex or efkreq.cmd
 - efkserv.rex or efkserv.cmd
 - efklist.rex or efklist.cmd
 - efkgen.ini
 - abt.cnf
 - hpt.ini

This file has been tailored to include the VisualAge Generator environment variables and to point to the repository.

 - abt.icx

This is an workspace/image that already has the VisualAge Generator Developer feature loaded.
7. Copy and name the efkgen.ini file in d:\langen\cmds as the following files:
 - efkgensv.ini

Tailor this file to reflect the drives that the generation server uses. For example, set the following: `/server_input_dir=d:\langen\inputgen` and `/gen_listing_dir=d:\langen\outlist`

- `efkgencl.ini`

Tailor this file to reflect the drives that the client uses. For example, set the following: `/server_input_dir=h:\input` and `/gen_listing_dir=h:\outlist`

8. Make your changes in the copies of `efkserv` and `efkreq` in `d:\langen\cmds`, not in `d:\vast`. This ensures that any `fixpak` changes to `efkserv` and `efkreq` do not destroy your changes.
9. Copy the templates from `\vast\template` to `d:\langen\template` and then make any tailoring changes you need to the version in `d:\langen\template`. Be sure to point your generation options to the tailored version of the templates.
10. Set the `GENERATE_INI` environment variable to point to `d:\langen\cmds\efkgensv.ini`
11. Set the `PATH` environment variable to include `d:\langen\cmds` before the `d:\vast` directory.
12. If you are running on a Novell Netware LAN environment, set the `EZERUSRID` environment variable to the LAN userid of the generation server.
13. Reboot your system for the changes to take effect.
14. Start `EFKSERV` from a command prompt.
15. For Windows NT, ensure that the date style and the time style in the Regional Settings match the corresponding information on the repository.

Setup on the client

The following example assumes that VisualAge Generator is installed on a LAN server and that all clients install VisualAge Generator on the same directory and use the same mappings of drives on the file server. You must complete the following tasks on the clients:

1. Install VisualAge Smalltalk client (directory `d:\vast`)
2. Install VisualAge Generator Developer
3. For Windows NT, install and tailor Object REXX
4. Map the generation server drive so that drive `h:` on the client workstation points to `d:\langen` on the server machine.
5. Set the `GENERATE_INI` environment variable to point to `h:\cmds\efkgencl.ini`
6. Set the `PATH` environment variable to include `h:\cmds` before the `d:\vast` directory.

This is needed so you can use `EFKREQ` from the generation server.

7. If you are running on a Novell Netware LAN environment, set the EZERUSRID environment variable to the LAN userid of the client workstation.

For Windows NT, this should be a user environment variable because it would vary depending on which user is signed on.

8. From the VisualAge Organizer window, select **Options** then **Preferences**. Then, from the **VAGen - Generation** tab, select **CALL EFKREQ** as the Batch generation command.
9. Reboot your system for the changes to take effect.
10. For Windows NT, ensure that the date style and the time style in the Regional Settings match the corresponding information on the repository.

Appendix C. Reading syntax diagrams

The syntax diagrams used throughout the documentation conform to the following conventions:

- The keywords can be listed in any order.

The \blacktriangleright — symbol indicates the beginning of a statement.

The — \blacktriangleright symbol indicates that the statement syntax is continued on the next line.

The \blacktriangleright — symbol indicates that a statement is continued from the previous line.

The — \blacktriangleleft symbol indicates the end of a statement.

A syntax diagram that does not show the complete statement starts with the \blacktriangleright — symbol and ends with the — \blacktriangleright symbol. The \blacktriangleright — \blacktriangleright symbol indicates the beginning of a statement.

A syntax diagram that show the complete statement starts with the \blacktriangleright — symbol and ends with the — \blacktriangleleft symbol.

- Required items appear on the horizontal line (the main path).



- Optional items appear below the main path.



- Items positioned above the syntax diagram line are default parameters.



- If you can choose from two or more items, these items appear vertically, in a stack.

If you *must* choose an item in the stack, one of the required items appears on the main path.



If choosing one of the items is optional, the entire stack appears below the main path.



- An arrow returning to the left above the item indicates an item that you can repeat. Required items appear on the main line and optional items appear below the main line.



A repeat arrow indicates that you can make more than one choice from the grouped items, or repeat a single item.

- Keywords appear in uppercase (for example, PARM1). However, they can be uppercase or lowercase when they are entered. They must be spelled exactly as shown. Variables and acceptable values appear in all lowercase letters (for example, parm x). They represent names or values that you supply.
- If punctuation marks, parentheses, arithmetic operators, or other symbols are shown, you must enter them as part of the syntax.

Index

Special Characters

<listener>.java.command
 java properties 363
<listener>.port
 java properties 363
<listener>.trace.flag
 java properties 363

A

adding a qualifier
 program user data set
 names 106
 to a preparation procedure 108
 to a preparation process
 template 113
 to a preparation template 107
 to a run-time CLIST
 template 111

AIX

 case sensitive 310
 example of generation 24
 slash in path name 215

AMODE, specifying 255

analyzing

 generation errors 367
 messages 165, 369
 preparation errors 18, 164, 368
 return codes 164, 368

ANSISQL option 300

appcims

 with calllink tag 224

applname attribute

 on calllink tag 220

assigning values to user-defined

 symbolic parameters 135

ASSOCIATE 267

 /BLKSIZE; 267
 /COMMIT 268
 /FILETYPE 269
 /NOCOMMIT 268
 /NODUP 268
 /NOREPLACE 271
 /PCBNO 271
 /REPLACE 271
 /SYSNAME 272
 /SYSNUM 272
 /SYSTEM 273
 FILE 267
 syntax 266

B

batch print services COBOL
 program 148

beans

 for record array rows 199
 for record parameters 198
 for servers 196

binary

 with calllink tag 227
 with ctxlink tag 236
 with filelink tag 241

binary table 151

BIND

 command file 257, 301
 command keywords 260
 command templates 71
 control part 55, 146, 257
 defining plans 257
 error return codes 264
 EXPLAIN(YES) keyword 260
 not required for OS/2, OS/400,
 or VSE environments 264
 OWNER(authorization-id)
 keyword 260
 sample commands 260

BIND option 301

binding

 packages instead of plans 262
 when the first program does not
 use SQL 262
 when the first program uses
 SQL 260

binform attribute

 on calllink tag 230

bitmode attribute

 on calllink tag 220

C

C++

 command interface 21
 GENERATE subcommand
 examples 24
 generating programs 5
 generation output 15
 inputs to program generation 11
 preparation process 17
 preparing programs 6
 C++ generation
 command interface 21

C++ generation (*continued*)

 GENERATE subcommand
 syntax 21
 inputs and outputs tables 11
 C++ preparation
 VALIDATE subcommand
 example 28, 47
 VALIDATE subcommand
 syntax 27, 47

ca400

 with calllink tag 224

CALL, effect on plans with RT
 generation option 259

CALL IDENTIFIER 88

call linkage 218

call linkage (CALLLINK) 218

CALL LITERAL 88

CALLLINK definition 220

calllink tag

binform attribute

 host 230
 INTEL 230

contable attribute

 binary 227
 conversion table name 226
 EZECONVT 226
 none 226

linktype attribute

 cicslink 221
 csocall 221
 dynamic 221
 remote 221
 sessionejb 222
 static 221

location attribute

 EZELOC 227
 system name 227

luwcontrol attribute

 client 229
 server 229

parmform attribute

 cicsoslink 224
 commdata 224
 commptr 223
 oslink 223

providerURL attribute

 URL 230

remotepptype attribute
 ITF 226

- calllink tag (*continued*)
 - NONVG 226
 - VG 225
 - VGJAVA 226
- remotebind attribute
 - generation 229
 - runtime 229
- remotecomtype attribute
 - appcims 224
 - ca400 224
 - cicsclient 224
 - dce 224
 - dcesecure 225
 - direct 225
 - exci 225
 - ipc 225
 - Java400 225
 - lu2 225
 - tcpip 225
- serverid attribute
 - server identifier 228
- case sensitive, generation
 - options 215
- cataloged procedure 83
- CHECKTYPE option 302
- cics
 - with crtlink tag 236
 - with filelink tag 242
- CICS
 - file types supported 276
 - parameter format and linkage combinations 231
 - required conversion options 86
 - TEMPAUX 279
 - templates for table definition 73
 - templates for tables 72
 - TEMPMAIN 280
 - TRANSIENT 281
- CICS/ESA, DBCS translator option
 - for 87
- CICS for AIX
 - SEQ 277
 - VSAM file type 282
- CICS for MVS/ESA
 - DBRMs 259
 - file type keywords 283
 - generation example 171
 - inputs and outputs for COBOL generation 51
 - naming program plans 258
 - SPOOL 278
- CICS for OS/2
 - creating COBOL compile and link listings for 118
 - generation example 173
- CICS for OS/2 (*continued*)
 - initializing the environment
 - for 119
 - inputs and outputs for COBOL generation 51
 - OS2COBOL 276
 - outputs of preparation 163
 - preparation 162
 - PREPARE subcommand 176
 - suppressing CICS translator, COBOL compile, and link messages 119
 - table entries 163
 - VSAM file type keyword 283
- CICS for VSE/ESA
 - file type keywords 283
 - generation example 174
 - inputs and outputs for COBOL generation 51
 - map group preparation
 - templates 66
 - preparation templates 64
 - SPOOL 278
- CICS for Windows NT
 - SEQ 277
 - VSAM file type 282
- CICS table templates 72
- cicsclient
 - with calllink tag 224
- CICSDBCS option 303
- CICSENTRIES option 303
- cicslink
 - with calllink tag 221
- cicoslink
 - with calllink tag 224
- client
 - with calllink tag 229
- client generation xv
- CLIST templates for MVS/TSO 79
- COB2LIB symbolic parameter 131
- COBLIST symbolic parameter 131
- COBOL
 - compiler options for MVS 88
 - copybook for MFS MID/MOD layout 150
 - creating compile and link listings
 - for CICS for OS/2 118
 - deleting 114
 - GENERATE subcommand
 - syntax 167
 - generating programs 5
 - inputs and outputs for generation 51
 - inputs to generation 51
 - link-editing static calls 245
- COBOL (*continued*)
 - outputs of generation 137
 - preparation process 153
 - preparing programs 6
 - program 144
 - run-time options for VSE 95
 - syntax for the VALIDATE subcommand 177
 - table program 151
 - VisualAge Generator Developer commands 167
- COBOL generation
 - outputs 137
 - templates used in 57
- COBOL program 144
- command interface
 - C++ generation 21
 - Java generation 43, 187, 201
- command keywords, BIND 260
- commands, options 297
- commdata
 - with calllink tag 224
- COMMENTLEVEL
 - details for C++ 305
 - details for COBOL 304
- common preparation errors 18
- commpr
 - with calllink tag 223
- compiler, RES option 89
- compiler options
 - additional 89
 - included 89
 - required for COBOL 88
 - that are not supported 91
- CONFIGMAPNAME
 - parameter 298
- CONFIGMAPVERSION
 - parameter 299
- considerations for plan definition 258
- contable attribute
 - on calllink tag 226
 - on crtlink tag 235
 - on filelink tag 240
- CONTABLE option 306
- contacting IBM for support 371
- control files, COBOL generation 51
- control statements, specifying 248, 252
- CONVERSE, effect on plans with RT generation option 259
- conversion table name
 - with calllink tag 226
 - with crtlink tag 236
 - with filelink tag 241

- conversion tables 13, 33, 55
- CREATEDDS option 306
- creating
 - a resource association file 265
 - COBOL compile and link listings for CICS for OS/2 118
 - generation options files 209
 - user-defined symbolic parameters 134
- creating linkage table entries 217
- CREATX linkage (CRTXLINK) 234
- crtxlink tag
 - countable attribute
 - binary 236
 - conversion table name 236
 - EZECONVT 236
 - linktype attribute
 - local 235
 - remote 235
 - location attribute
 - cics 236
 - EZELOC 237
- cso.application.<applname>
 - java properties 365
- cso.serverLinkage.<group>.<attribute>
 - java properties 365
- csocall
 - with calllink tag 221
- CSOTcpipListener 362
- CSOUiListener 362
- CURRENCY option 306
- D**
- DATA option 306
- DB2, required options 86
- DB2/VSE
 - naming program packages 95
 - required options 94
- DBCS
 - compiler option 89
- DBCS translator option for CICS/ESA 87
- DBDLIB symbolic parameter 131
- DBMS option 307
- DBPASSWORD option 308
- DBUSR option 308
- dce
 - with calllink tag 224
- dcsecure
 - with calllink tag 225
- DEBUGTRACE option 308
- default
 - establishing generation options 210
 - generation options parts examples 211
- default (*continued*)
 - generation options parts when NOOVERRIDE is not specified 211
 - generation options parts when NOOVERRIDE is specified 210
 - reserved-word file 54
 - sample generation options parts 213
 - template extension 57
- defining
 - BIND plans 257
 - link-edit file 245
- definitions for CALLLINK 220
- deleting COBOL on MVS or VSE host 114
- deleting COBOL source from the workstation 113
- DESTACCOUNT option 309
- DESTDIR option 309
- DESTHOST option 310
- DESTLIB option 310
- DESTPASSWORD option 310
- DESTUID option 311
- detail 319
 - /GROUPNAME 319
- determining generation option resolution order 212
- diagram
 - CICS for MVS/ESA generation 51
 - CICS for OS/2 generation 51
 - CICS for VSE/ESA generation 51
 - COBOL generation 51
 - IMS BMP COBOL generation 51
 - IMS VS COBOL generation 51
 - inputs and outputs for C++ generation 11
 - MVS batch generation 51
 - MVS TSO generation 51
 - OS/400 generation 51
 - VSE batch generation 51
- different systems, preparing programs on 85
- direct
 - with calllink tag 225
- DL/I
 - usage 79
 - usage for MVS batch 63, 76
 - usage for MVS/TSO 63
 - VSE batch 78
- DSNLOAD symbolic parameter 132
- DSYS symbolic parameter 132
- DXFR, effect on plans with RT
 - generation option 259
- DXFR linkage (DXFRLINK) 237
- DXFRCANCEL 311
- dxfrlink tag
 - linktype attribute
 - dynamic 238
 - noncsp 239
 - static 238
- DXFRXCTL 311
- dynamic
 - with calllink tag 221
 - with dxfrlink tag 238
- E**
- effects of XFER, DXFR, CALL, CONVERSE, and the /RT generation option on plans 259
- EFK24PBC, preparation template 68
- EFK24PBM, preparation template 69
- EFK24PCL, preparation template 68
- EFK24PEC, preparation template 68
- EFK24PEJ, preparation template 70
- EFK24PMN, preparation template 68
- EFK24PPM, preparation template 68
- EFK24PSC, preparation template 68
- EFK24PSM, preparation template 68
- EFK24TAM, preparation template 69
- EFK24TCM, preparation template 69
- EFK24TEM, preparation template 69
- EFK24TMJ, preparation template 70
- EFK24WCL, preparation template 68
- EFK24WSC, preparation template 69
- EFK2CVCL template 82
- EFK2MDLI template 81
- EFK2MEBA template 78
- EFK2META template 80
- EFK2MIMS template 81
- EFK2MPCB template
 - before modification 120
 - preparation JCL 121
 - values for the symbolic parameters 120
- EFK2MTCL template 82
- EFK2MTDL template 81
- EFK2VDLI template 80

- EFK2VEBA template 78
 - EFKGEN.INI file
 - defines input queue directory 383
 - specifies server input queue 382
 - EFKREQ file 381
 - EJBGROUP option 311
 - ELA symbolic parameter 131
 - ELARLINK procedure,
 - modifying 248
 - ENDCOMMAREA option 312
 - Enterprise Java Bean (EJB)
 - generating 205
 - entries, required 298
 - ENTRY_POINT option 158
 - environment variable
 - PATHEXT 385
 - environment variables
 - HPTCLASSDIR 185
 - HPTJSPDIR 185
 - ERRDEST option 312
 - error return code, BIND
 - commands 264
 - error return codes, static links 256
 - errors
 - analyzing generation 367
 - analyzing preparation 164, 368
 - common preparation 18
 - preparation 18
 - establishing default generation
 - options 210
 - example
 - default generation options
 - files 211
 - GENERATE subcommand 171
 - GENERATE subcommand for
 - C++ 24
 - GENERATE subcommand for
 - Java 189
 - generating for AIX 24
 - generating for CICS for
 - MVS/ESA 171
 - generating for CICS for
 - OS/2 173
 - generating for CICS for
 - VSE/ESA 174
 - generating for HP-UX 24
 - generating for MVS batch 172
 - generating for MVS/TSO 172
 - generating for OS/2 25
 - generating for SCO 25
 - generating for Solaris 25
 - generating for VM batch 174
 - generating for VM CMS 173
 - generating for VSE batch 174
 - example (*continued*)
 - generating for Windows NT 26, 45
 - modifying templates 106
 - PREPARE subcommand 176
 - VALIDATE subcommand 179
 - VALIDATE subcommand for
 - C++ 28, 47
 - VALIDATE subcommand for
 - Java 204
 - exci
 - with calllink tag 225
 - extension
 - templates 57
 - externalname attribute
 - on calllink tag 223
 - EZEBLK symbolic parameter 130
 - EZECOBOLTYPE, symbolic
 - parameter 124
 - EZECONVT
 - with calllink tag 226
 - with crtlink tag 236
 - with filelink tag 241
 - EZEDATA symbolic parameter 124
 - EZEDBCS symbolic parameter 124
 - EZEDBD symbolic parameter 130
 - EZEDD symbolic parameter 130
 - EZEDESTLIB symbolic
 - parameter 124
 - EZEDESTNAME symbolic
 - parameter 124
 - EZEDLBL symbolic parameter 130
 - EZEDLI symbolic parameter 124
 - EZEEDSN symbolic parameter 131
 - EZEENTRY symbolic
 - parameter 124
 - EZEENV symbolic parameter 124
 - EZEGDATE symbolic
 - parameter 125
 - EZEGENOUT, symbolic
 - parameter 125
 - EZEGMBR symbolic parameter 125
 - EZEGTIME symbolic
 - parameter 125
 - EZELOC
 - with calllink tag 227
 - with crtlink tag 237
 - with filelink tag 242
 - EZELRECL symbolic parameter 131
 - EZEMBR symbolic parameter 125
 - EZEMBRPATH symbolic
 - parameter 125
 - EZEMSG symbolic parameter 125
 - EZEPID symbolic parameter 125
 - EZEPREPDESTACCOUNT 125
 - EZEPREPDESTDIR 126
 - EZEPREPDESTHOST 126
 - EZEPREPDESTEPASSWORD 126
 - EZEPREPDESTUID 126
 - EZEPREPFTPCMDDBCS 126
 - EZEPREPFTPCMDSBCS 126
 - EZEPREPSENDICMDDBCS 126
 - EZEPREPSESSION 127
 - EZEPREPSP 127
 - EZEPREPSQLDB 127
 - EZEPREPWORKDB 127
 - EZEPSB symbolic parameter 127
 - EZEPTYPE part types 127
 - EZEPTYPE symbolic parameter 127
 - EZERECFM symbolic
 - parameter 131
 - EZESQL symbolic parameter 128
 - EZETRAN symbolic parameter 129
 - EZETRANSFERTYPE 129
 - EZEUSRID environment
 - variable 91
 - EZEUSRID symbolic parameter 129
 - EZEVELIB symbolic
 - parameter 129
 - EZEXAPP symbolic parameter 129
 - EZUAUTH symbolic parameter 132
 - EZUINST symbolic parameter 132
- ## F
- FASTPATH option 313
 - FCEJBLD
 - FCEJBLD 39
 - file and database allocation
 - placeholder templates 81
 - file and database allocation
 - templates 80
 - file linkage 239
 - FILE linkage (FILELINK) 239
 - file-related symbolic
 - parameters 130
 - file type
 - OS2COBOL 276
 - SEQ 277
 - SPOOL 278
 - supported by CICS
 - environments 276
 - supported by environment and
 - record organization 274
 - supported by OS/400 289
 - supported for MVS Batch,
 - IMS/Vs, and IMS BMP 287
 - supported for MVS/TSO 286
 - supported for VM
 - environments 290
 - supported for VSE batch 292
 - TEMPAUX 279

- file type *(continued)*
 - TEMPMAIN 280
 - TRANSIENT 281
 - VSAM 283
 - VSAM for CICS for AIX< 282
 - VSAM for CICS for Windows NT 282
- filelink tag
 - contable attribute
 - binary 241
 - conversion table name 241
 - EZECONVT 241
 - linktype attribute
 - local 240
 - remote 240
 - location attribute
 - cics 242
 - EZELOC 242
- filename attribute
 - on filelink tag 240
- filename option 298
- FOLD option 313
- fromappl attribute
 - on dxfrlink tag 237
- FTPTRANSLATIONCMDDBCS 313
- FTPTRANSLATIONCMDSBCS 314

G

- GENAUTHORTIMEVALUES
 - option 314
- GENERATE command
 - Java 43
- GENERATE subcommand 297
 - examples 171
 - examples for C++ 24
 - examples for Java 189
 - return codes 367
 - syntax for C++ 21
 - syntax for COBOL 167
 - syntax for Java 187, 201
- generation
 - analyzing errors 367
 - COBOL and C++ 5
 - command interface for C++ 21
 - command interface for Java 187, 201
 - inputs and outputs for C++ 11
 - inputs to C++ 11
 - inputs to Java 181, 193
 - inputs to Java server program
 - generation 31
 - inputs to Java wrappers 193
 - introducing 3
 - modifying output 151
 - on a LAN server, running 381

- generation *(continued)*
 - options for each
 - environment 375
 - options for Java wrappers 193
 - output for C++ 15
 - output for java 35
 - output for Java 183
 - output for Java wrappers 195
 - sample option files 213
 - templates used for COBOL 57
 - with calllink tag 229
- generation binding for linkage
 - options 194
- generation examples
 - AIX 24
 - CICS for MVS/ESA 171
 - CICS for OS/2 173
 - CICS for VSE/ESA 174
 - GENERATE subcommand for C++ 24
 - GENERATE subcommand for Java 189
 - HP-UX 24
 - MVS batch 172
 - MVS/TSO 172
 - OS/2 25
 - SCO 25
 - Solaris 25
 - VM batch 174
 - VM CMS 173
 - VSE batch 174
 - Windows NT 26, 45
- generation option
 - /ANSISQL 300
 - /BIND 301
 - /CHECKTYPE 302
 - /CICSDBCS 303
 - /CICSENTRIES 303
 - /COMMENTLEVEL for C++ 305
 - /COMMENTLEVEL for COBOL 304
 - /CONFIGMAPNAME 298
 - /CONFIGMAPVERSION 299
 - /CONTABLE 306
 - /CREATEDDS 306
 - /CURRENCY 306
 - /DATA 306
 - /DBMS 307
 - /DBPASSWORD 308
 - /DBUSER 308
 - /DEBUGTRACE 308
 - /DESTACCOUNT 309
 - /DESTDIR 309
 - /DESTHOST 310

- generation option *(continued)*
 - /DESTLIB 310
 - /DESTPASSWORD 310
 - /DESTUID 311
 - /EJBGROUP 311
 - /ENDCOMMAREA 312
 - /ERRDEST 312
 - /FOLD 313
 - /FTPTRANSLATIONCMDDBCS 313
 - /FTPTRANSLATIONCMDSBCS 314
 - /GENAUTHORTIMEVALUES 314
 - /GENHELPMAPS 314
 - /GENMAPS 315
 - /GENOUT 315
 - /GENRESOURCEBUNDLE 317
 - /GENRET 318
 - /GENTABLES 319
 - /GENUIRECORDS 319
 - /INEDIT 320
 - /INITADDWS 320
 - /INITRECD 321
 - /JAVADESTDIR 321
 - /JAVADESTHOST 322
 - /JAVADESTPASSWORD 322
 - /JAVADESTUID 323
 - /JAVASYSTEM 323
 - /JOB CARD 323
 - /JOBNAME 324
 - /JSPRELDIR 324
 - /LEFTJUST 324
 - /LIB 355
 - /LINEINFO 325
 - /LINES 325
 - /LINKAGE 326
 - /LINKEDIT 326
 - /LISTING 326
 - /LISTINGONERROR 326
 - /LOCVALID 327
 - /LOG 327
 - /MATH 327
 - /MFSDEV 328
 - /MFSEATTR 329
 - /MFSEATTRNCD 329
 - /MFSIGNORE 329
 - /MFSTEST 330
 - /MSGTABLEPREFIX 330
 - /MSP 330
 - /NOANSISQL 331
 - /NOCICSDBCS 331
 - /NODEBUGTRACE 308
 - /NOENDCOMMAREA 312
 - /NOFASTPATH 313
 - /NOFOLD 313
 - /NOGENHELPMAPS 314
 - /NOGENMAPS 315

generation option (*continued*)

/NOGENRET 318
/NOGENTABLES 319
/NOINITADDWS 320
/NOINITRECD 321
/NOLEFTJUST 324
/NOLINEINFO 325
/NOLISTING 326
/NOLISTING, /LISTING,
/LISTINGONERROR 326
/NOLOCVALID 327
/NOLOG 327
/NOMFSEATTR 329
/NOMFSIGNORE 329
/NOMFSTEST 330
/NONULLFILL 334
/NONUMOVFL 334
/NOPREP 335
/NOPREPROFILE 336
/NORECOVERY 337
/NORUNFILE 342
/NOSETFULL 342
/NOSPZERO 343
/NOSQLVALID 345
/NOSYNCDXFR 347
/NOSYNCFER 347
/NOSYSCODES 348
/NOUNLOAD 333
/NOVALIDMIX 354
/NULLFILL 334
/NUMOVFL 334
/OPTIONS 334
/PACKAGENAME 335
/POSSIGN 335
/PREP 335
/PREPROFILE 336
/PRINTDEST 336
/PROJECT 299
/PROJECTID 337
/RECOVERY 337
/RESOURCE 338
/RESOURCEBUNDLELOCALE 339
/RESVWORD 340
/RT 341
/RUNFILE 342
/SENDTRANSLATIONCMDDBCS 342
/SESSION 342
/SETFULL 342
/SP 343
/SPA 343
/SPZERO 343
/SQLDB 344
/SQLID 345
/SQLPASSWORD 345
/SQLVALID 345

generation option (*continued*)

/SYMPARM 346
/SYNCDXFR 347
/SYSCODES 348
/SYSTEM 299
/TARGNLS 348
/TEMPLATES 350
/TRACE 350
/TRANSFERTYPE 351
/TRANSID 352
/TWOOFF 353
/UNLOAD 353
/VALIDMIX 354
/VMLOADLIB 354
/WORKDB 356
case sensitive options 215
commands details 297
database management
 system 307
default files 211
details 297
determining resolution
 order 212
DXFRCANCEL 311
DXFRXCTL 311
establishing defaults 210
filename 298
GENERATE 297
guidelines for setting 215
HPTCMD 297
HPTCMD details 297
naming conventions 215
NOOVERRIDE not specified 211
NOOVERRIDE specified 210
not valid 215
options details 300
overriding a value to use
 default 215
 partname 298
PREPARE 297
reading 389
 subcommand details 297
 symbolic parameters 214
 VALIDATE 297
generation options for each
 environment 375
generation options part
 available during generation 210
 creating 209
 description 209
 multiple levels 212
 sample 213
generation outputs 195
generation server
 setting up 385

GENHELPMAPS option 314

GENMAPS option 315
GENOUT option 315
GENRESOURCEBUNDLE
 option 317
GENTABLES option 319
GENUIRECORDS option 319
getting ready for OS/400
 preparation 162
GROUPNAME option 319
guideline, setting generation
 options 215

H

host
 with calllink tag 230
host services
 DBRMs for CICS for
 MVS/ESA 259
HP-UX
 case sensitive 310
 example of generation 24
 slash in path name 215
HPTCMD details 297

I

IBM, contacting for support 371
IMS, naming program plans 259
IMS BMP
 file types supported 287
 inputs and outputs for COBOL
 generation 51
 map group preparation
 templates 66
 preparation templates 64
 run-time JCL templates 76
IMS/VS
 DBRMs 260
 file types supported 287
 inputs and outputs for COBOL
 generation 51
 map group preparation
 templates 66
 preparation templates 64
IMS/VS DBRMs 260
included compiler options 89
INEDIT option 320
INITADDWS option 320
initializing the environment for CICS
 for OS/2 119
INITRECD option 321
inputs
 C++ program generation 11
 COBOL generation 51
 Java program generation 181

- inputs (*continued*)
 - Java server program
 - generation 31
- inputs and outputs tables
 - C++ generation 11
 - COBOL generation 51
 - COBOL generation, CICS for
 - MVS/ESA 51
 - COBOL generation, CICS for
 - OS/2 51
 - COBOL generation, CICS for
 - VSE/ESA 51
 - COBOL generation, IMS
 - BMP 51
 - COBOL generation, IMS VS 51
 - COBOL generation, MVS
 - batch 51
 - COBOL generation, MVS
 - TSO 51
 - COBOL generation, OS/400 51
 - COBOL generation, VSE
 - batch 51
 - Java wrapper generation 193
- INTEL
 - with calllink tag 230
- interfaces requiring a linkage
 - table 232, 239
- interfaces using CALL or DXFR
 - statements on a single
 - system 232, 239
- introducing program generation 3
- ipc
 - with calllink tag 225
- issuing VisualAge Generator
 - Developer commands 167
- ITF
 - with calllink tag 226
- J**
- java
 - generation output for 35
- Java
 - command interface 187, 201
 - GENERATE subcommand
 - examples 189
 - GENERATE subcommand
 - syntax 187, 201
 - generation output 183
 - inputs to program
 - generation 181, 193
 - inputs to server program
 - generation 31
 - preparation 39
 - preparation process 185
- Java generation
 - command interface 43, 187, 201
 - GENERATE command 43
 - inputs 181
- Java generation outputs 183
- Java Naming and Directory Interface
 - (JNDI) 205
- Java preparation
 - fcjblid 39
 - VALIDATE subcommand
 - example 204
 - VALIDATE subcommand
 - syntax 204
- java properties
 - <listener>.java.command 363
 - <listener>.port 363
 - <listener>.trace.file 363
 - <listener>.trace.flag 363
 - applname 365
 - cso.application.<applname> 365
 - cso.linkagetable.<linktable> 364
 - cso.serverLinkage.<group>.<attribute> 363
 - CSOTcpipListener 362
 - CSOUiListener 362
 - linkage table 364
 - server group 365
- Java properties
 - contable 360
 - database default 361
 - decimal 364
 - default database 361
 - filetype 359
 - java command 362
 - JDBC drivers 361
 - JDBC server name 362
 - JVM command 362
 - linkage 364
 - listener 362
 - NLS 364
 - nls code 364
 - replace 359
 - resource association 359
 - SQL password 361
 - SQL user ID 361
 - sysname 360
 - text 360
- Java wrapper generation
 - inputs 193
- Java400
 - with calllink tag 225
- JavaBeans wrappers
 - generating 191
 - generation outputs 195
 - generation inputs 193
 - generation options 193
- JavaBeans wrappers (*continued*)
 - generation output 195
 - inputs to program
 - generation 193
 - linkage table options 194
 - JVADESTDIR option 321
 - JVADESTHOST option 322
 - JVADESTPASSWORD option 322
 - JVADESTUID option 323
 - JAVASYSTEM option 323
 - JCL templates 75
 - JOB statements 74
 - JOBCARD option 323
 - JOBNAME option 324
 - JSPRELDIR option 324
- L**
- LAN generation
 - setting up 385
- LE
 - modifying user exits 87
 - sub-time options 87
- LEFTJUST option 324
- library attribute
 - on calllink tag 220
- LINEINFO option 325
- LINES option 325
- link-edit
 - defining a file 245
 - link-editing static COBOL
 - calls 245
 - linkage editor control statement
 - files 53, 245
 - linkage editor control statements
 - for MVS 246
 - linkage editor control statements
 - for VM 250
- link-edit templates 71
- link-editing, static COBOL calls 245
- linkage
 - table entries
 - creating 217
 - sample 242
 - table entry format 217
 - tables 217
- linkage editor control statements
 - files 53, 245
 - MVS 246
 - VM 250
 - VSE 253
- LINKAGE option 326
- linkage table
 - cso.linkagetable.<linktable> 364
 - java properties 364
 - linkage table description 53

- linkage table options
 - for Java wrappers 194
 - generation binding 194
 - run-time binding 194
- LINKEDIT option 326
- linktype attribute
 - on calllink tag 221
 - on crtlink tag 235
 - on dxfrlink tag 238
 - on filelink tag 240
- list of files
 - transferred to MVS 161
 - transferred to OS/400 162
 - transferred to VSE 161
- list of terms used in this document xii
- listing file 141
- LISTING option 326
- LISTINGONERROR option 326
- local
 - with crtlink tag 235
 - with filelink tag 240
- location attribute
 - on calllink tag 227
 - on crtlink tag 236
 - on filelink tag 241
- LOCVALID option 327
- LOG option 327
- lu2
 - with calllink tag 225
- luwcontrol attribute
 - on calllink tag 229
- M**
- map group
 - format module 148
 - templates 65
- map group template
 - CICS for VSE/ESA and VSE batch preparation 66
 - IMS/VS and IMS BMP preparation 66
- MATH option 327
- member-related symbolic parameters 123
- messages, analyzing 165, 369
- MFS
 - COBOL copybook for MID/MOD layout 150
 - control blocks 149
 - print services COBOL program 149
- MFSDEV option 328
- MFSEATTR option 329
- MFSEATTRNCD option 329
- MFSIGNORE option 329
- MFSTEST option 330
- modifying
 - ELARLINK procedure 248
 - generation output 151
 - LE user exits 87
 - MVS JOB card 91
 - PSB name to match batch program name 115
 - templates 82
 - templates and procedures for MVS environments 84
 - templates for the OS/400 environment 92
 - templates for VM 97
 - templates for VSE 93
 - the EFK2MPCB template 120
 - VSE JOB card 95
- MSGTABLEPREFIX option 330
- MSP option 330
- multiple resource associations for a file 266
- MVS
 - COBOL compiler options 88
 - deleting COBOL 114
 - getting ready for preparation 160
 - linkage editor control statements 246
 - list of files transferred 161
 - modifying templates and procedures 84
 - modifying the JOB card 91
 - preparation 153
 - preparation file templates 62
 - preparation JCL 142
 - preparation script templates 58
 - PREPARE subcommand 176
 - program and transaction definitions 143
 - program with static calls to other programs 246
 - programs statically called by other programs 247
 - setting COBOL run-time options 87
- MVS batch
 - DBRMs 259
 - DL/I usage 63, 76
 - file types supported 287
 - generation example 172
 - inputs and outputs for COBOL generation 51
 - naming program plans 258
 - MVS preparation 160
- MVS/TSO
 - CLIST templates 79
 - DBRMs 259
 - DL/I usage 63
 - file types supported 286
 - generation example 172
 - inputs and outputs for COBOL generation 51
 - naming program plans 258
 - routing output 116
- N**
- names, generation options files 213
- naming
 - CICS for MVS/ESA program plans 258
 - conventions for allocated data sets 160
 - DB2/VSE program packages 95
 - IMS program plans 259
 - MVS/TSO and MVS batch program plans 258
- NET RUN 381
- NLS codes 349
- NOANSISQL option 331
- NOCICSDBCS option 331
- NODEBUGTRACE option 308
- NODYNAM compiler option 88
- NOENDCOMMAREA option 312
- NOFASTPATH option 313
- NOFOLD option 313
- NOGENHELPMAPS option 314
- NOGENMAPS option 315
- NOGENRET option 318
- NOGENTABLES option 319
- NOINITADDWS option 320
- NOINITRECD option 321
- NOLEFTJUST option 324
- NOLINEINFO option 325
- NOLISTING option 326
- NOLOCVALID option 327
- NOLOG option 327
- NOMFSEATTR option 329
- NOMFIGNORE option 329
- NOMFSTEST option 330
- noncsp
 - with dxfrlink tag 239
- none
 - with calllink tag 226
- NONULLFILL option 334
- NONUMOVFL option 334
- NONVG
 - with calllink tag 226

NOOVERRIDE
 in default generation options
 file 210
 not in default generation options
 file 211
 NOPREP option 335
 NOPREPROFILE option 336
 NORECOVERY option 337
 NOREPLACE, ASSOCIATE 271
 NORUNFILE option 342
 NOSPZERO option 343
 NOSQLVALID option 345
 NOSYNCDXFR option 347
 NOSYNCFER option 347
 NOSYCODES option 348
 NOUNLOAD option 333
 NOVALIDMIX option 354
 NULLFILL option 334
 NUMOVFL option 334

O

Object REXX support for
 Windows NT 384
 objects generated
 all member types 138
 map groups 147
 programs 144
 tables 150
 online print services COBOL
 program 149
 option
 DBCS compiler 89
 details 300
 establishing default
 generation 210
 NODYNAM compiler 88
 RES compiler 89
 options
 /ANSISQL 300
 /BIND 301
 /CHECKTYPE 302
 /CICSDBCS 303
 /CICSENTRIES 303
 /COMMENTLEVEL for
 C++ 305
 /COMMENTLEVEL for
 COBOL 304
 /CONTABLE 306
 /CREATEDDS 306
 /CURRENCY 306
 /DATA 306
 /DBPASSWORD 308
 /DBUSER 308
 /DEBUGTRACE 308
 /DESTACCOUNT 309
 /DESTDIR 309

options (*continued*)
 /DESTHOST 310
 /DESTLIB 310
 /DESTPASSWORD 310
 /DESTUID 311
 /EJBGROUP 311
 /ENDCOMMAREA 312
 /ERRDEST 312
 /FOLD 313
 /GENAUTHORTIMEVALUES 314
 /GENHELPMAPS 314
 /GENMAPS 315
 /GENOUT 315
 /GENRESOURCEBUNDLE 317
 /GENRET 318
 /GENTABLES 319
 /GENUIRECORDS 319
 /INEDIT 320
 /INITADDWS 320
 /INITRECD 321
 /JVADESTDIR 321
 /JVADESTHOST 322
 /JVADESTPASSWORD 322
 /JAVASYSTEM 323
 /JOB CARD 323
 /JOBNAME 324
 /JSPRELDIR 324
 /LEFTJUST 324
 /LIB 355
 /LINEINFO 325
 /LINES 325
 /LINKAGE 326
 /LINKEDIT 326
 /LOCVALID 327
 /LOG 327
 /MATH 327
 /MFSDEV 328
 /MFSEATTR 329
 /MFSEATTRNCD 329
 /MFSIGNORE 329
 /MFSTEST 330
 /MSGTABLEPREFIX 330
 /MSP 330
 /NOANSISQL 331
 /NOCICSDBCS 331
 /NODEBUGTRACE 308
 /NOENDCOMMAREA 312
 /NOFASTPATH 313
 /NOFOLD 313
 /NOGENHELPMAPS 314
 /NOGENMAPS 315
 /NOGENRET 318
 /NOGENTABLES 319
 /NOINITADDWS 320
 /NOINITRECD 321

options (*continued*)
 /NOLEFTJUST 324
 /NOLINEINFO 325
 /NOLOCVALID 327
 /NOLOG 327
 /NOMFSEATTR 329
 /NOMFIGNORE 329
 /NOMFSTEST 330
 /NONULLFILL 334
 /NONUMOVFL 334
 /NOPREP 335
 /NOPREPROFILE 336
 /NORECOVERY 337
 /NORUNFILE 342
 /NOSETFULL 342
 /NOSPZERO 343
 /NOSQLVALID 345
 /NOSYNCDXFR 347
 /NOSYNCFER 347
 /NOSYCODES 348
 /NOUNLOAD 333
 /NOVALIDMIX 354
 /NULLFILL 334
 /NUMOVFL 334
 /OPTIONS 334
 /PACKAGENAME 335
 /POSSIGN 335
 /PREP 335
 /PREPROFILE 336
 /PRINTDEST 336
 /PROJECT 299
 /PROJECTID 337
 /RECOVERY 337
 /RESOURCE 338
 /RESOURCEBUNDLELOCALE 339
 /RESVWORD 340
 /RT 341
 /RUNFILE 342
 /SESSION 342
 /SETFULL 342
 /SP 343
 /SPA 343
 /SPZERO 343
 /SQLDB 344
 /SQLID 345
 /SQLVALID 345
 /SYMPARM 346
 /SYNCDXFR 347
 /SYSCODES 348
 /SYSTEM 299
 /TARGNLS 348
 /TEMPLATES 350
 /TRACE 350
 /TRANSID 352
 /TWAOFF 353

- options (*continued*)
 - /UNLOAD 353
 - /VALIDMIX 354
 - /VMLOADLIB 354
 - /WORKDB 356
 - additional compiler 89
 - CICS required conversion 86
 - commands 297
 - compiler options that are not supported 91
 - DBCS translator for
 - CICS/ESA 87
 - filename 298
 - for COBOL compiler for
 - MVS 88
 - for setting COBOL run-time for
 - MVS 87
 - GENERATE 297
 - HPTCMD 297
 - included compiler 89
 - LE run-time 87
 - options 300
 - partname 298
 - PREPARE 297
 - required 88
 - required DB2 86
 - required for DB2/VSE 94
 - setting additional SQL preprocessor 94
 - setting COBOL run-time for
 - VSE 95
 - subcommands 297
 - syntax diagram 297
 - VALIDATE 297
 - OPTIONS option 334
 - OS/2
 - BIND command files not required 264
 - example of generation 25
 - OS/400
 - BIND command files not required 264
 - CICS for OS/2 176
 - example 176
 - file types supported 289
 - getting ready for
 - preparation 162
 - inputs and outputs for COBOL
 - generation 51
 - list of files transferred 162
 - modifying templates for the
 - MVS 176
 - OS/400 176
 - preparation 161
 - PREPARE subcommand 176
 - OS/400 (*continued*)
 - return codes 368
 - syntax for COBOL 175
 - templates 67
 - VSE 176
 - OS2COBOL, file type keyword for
 - the CICS for OS/2 environment 276
 - oslink
 - with calllink tag 223
 - output
 - C++ generation 15
 - COBOL generation 137
 - java generation 35
 - Java generation 183
 - Java wrapper generation 195
 - modifying generation 151
 - outputs, CICS for OS/2
 - preparation 163
 - overriding a value to use the default value 215
 - overview about generation control files 51
- P**
- package attribute
 - on crtlink tag 235
 - PACKAGENAME option 335
 - parameters
 - format and linkage combinations for CICS 231
 - parmform attribute
 - on calllink tag 223
 - partname option 298
 - PATHEXT environment
 - variable 385
 - PCT, program control table 73
 - POSSIGN option 335
 - PPT, program properties table 73
 - predefined symbolic
 - parameters 123
 - PREP option 335
 - preparation
 - analyzing errors 18, 164, 368
 - C++ 17
 - CICS for OS/2 162
 - command file templates 58
 - common errors 18
 - customizing 159
 - for COBOL 153
 - for MVS and VSE systems 153
 - getting ready for MVS 160
 - getting ready for OS/400 162
 - getting ready for VSE 161
 - Java 39, 185
 - preparation (*continued*)
 - JCL for MVS and VSE environments 142
 - MVS and VSE 153
 - OS/400 161
 - REXX exec for VM environments 143
 - VALIDATE subcommand
 - example for C++ 28, 47
 - VALIDATE subcommand
 - example for Java 204
 - VALIDATE subcommand syntax
 - for C++ 27, 47
 - VALIDATE subcommand syntax for Java 204
 - preparation command file 141
 - preparation file templates for MVS or VSE 62
 - preparation JCL, created from the EFK2MPCB template 121
 - preparation template
 - EFK24PBC 68
 - EFK24PBJ 69
 - EFK24PCL 68
 - EFK24PEC 68
 - EFK24PEJ 70
 - EFK24PMN 68
 - EFK24PPM 68
 - EFK24PSC 68, 69
 - EFK24PSM 68
 - EFK24TAM 69
 - EFK24TCM 69
 - EFK24TEM 69
 - EFK24TMJ 70
 - EFK24WCL 68
 - PREPARE subcommand 297
 - preparing
 - COBOL and C++ 6
 - programs on a single system 85
 - programs on different systems 85
 - PREPFILE option 336
 - PRINTDEST option 336
 - procedure name and symbolic
 - parameters 86
 - processing templates 120
 - PROCLIB symbolic parameter 132
 - program and transaction definitions, MVS and VSE 143
 - program control table (PCT) 73
 - program name
 - java properties 365
 - program properties table (PPT) 73
 - program templates 62

- program templates 62 (*continued*)
 - CICS for VSE/ESA and VSE
 - batch preparation 64
 - IMS/VS and IMS BMP
 - preparation 64
 - PROJECT option 299
 - PROJECTID option 337
 - providerURL attribute
 - on calllink tag 230
 - PSBLIB symbolic parameter 132
 - PWRCLASS symbolic parameter 132

Q

- qualifier
 - adding for program user data set names 106
 - adding to a preparation procedure 108
 - adding to a preparation process template 113
 - adding to a preparation template 107
 - adding to a run-time CLIST template 111

R

- reading syntax diagrams 389
- reasons to modify templates 83
- recdname attribute
 - on crtlink tag 235
- RECOVERY option 337
- remote
 - with calllink tag 221
 - with crtlink tag 235
 - with filelink tag 240
- remoteapptype attribute
 - on calllink tag 225
- remotebind attribute
 - on calllink tag 229
- remotecomtype attribute
 - on calllink tag 224
- REPLACE, ASSOCIATE 271
- required DB2 options 86
- required DB2/VSE options 94
- required options 88
- required parameters and options 298
- RES compiler option 89
- reserved-word file
 - contents 54
 - default 54
 - description 53
- RESLIB symbolic parameter 132
- resource association file
 - creating 265

- resource association file (*continued*)
 - description 265
 - multiple associations 266
 - sample 274
 - syntax 266
- Resource Associations Editor 265
- resource associations part 56, 265
- resource associations part when using VisualAge Generator 13, 32
- RESOURCE option 338
- RESOURCEBUNDLELOCALE option 339
- RESVWORD option 340
- return codes
 - analyzing 164, 368
 - GENERATE subcommand 367
 - PREPARE subcommand 368
- REXX 70
 - preparation exec for VM 70
 - setting up support for 384
- REXX exec 143
- REXXscript 384
- RMODE, specifying 255
- routing output, MVS/TSO 116
- RT option
 - details 341
 - effect on plans with XFER DXFR CALL CONVERSE 259
- run time
 - file templates 74
 - JCL templates for IMS BMP 76
 - JCL templates for VSE batch 77
 - REXX templates for VM 78
- run-time binding for linkage options 194
- run-time CLIST, sample 145
- run-time code
 - generating xv
- run-time JCL, sample 145
- run-time REXX exec, sample 146
- RUNFILE option 342
- running generation on a LAN server 381
- runtime
 - with calllink tag 229

S

- sample
 - BIND commands 260
 - default generation options files 213
 - generation options files 213
 - linkage table entries 242
 - resource association file 274
 - run-time CLIST 145

- sample (*continued*)
 - run-time JCL 145
 - run-time REXX exec 146
- sample files
 - EFKGEN.INI 382
 - EFKLIST 382
 - EFKREQ 382
 - EFKSERV 382
- SCO
 - case sensitive 310
 - example of generation 25
 - slash in path name 215
- SENDTRANSLATIONCMDDBCS option 342
- SEQ file-type keyword
 - for CICS for AIX 277
 - for CICS for Windows NT 277
- server
 - with calllink tag 229
- server group
 - java properties 365
- server identifier
 - with calllink tag 228
- serverid attribute
 - on calllink tag 228
- serviceability 371
- session bean
 - generating 205
- SESSION option 342
- sessionejb
 - with calllink tag 222
- SETFULL option 342
- setting additional SQL preprocessor options 94
- setting COBOL run-time options for MVS 87
- setting COBOL run-time options for VSE 95
- setting up
 - the client 382
 - the server 383
- single system, preparing programs on a 85
- Solaris
 - case sensitive 310
 - example of generation 25
 - slash in path name 215
- SP option 343
- SPA option 343
- specifying
 - AMODE and RMODE 255
 - control statements 248, 252
- SPOOL
 - file type keyword for CICS for MVS/ESA 278

- SPOOL (*continued*)
 - file type keyword for CICS for VSE/ESA environments 278
- SPZERO option 343
- SQL
 - preprocessing for VSE 93
 - setting additional preprocessor options 94
 - starting preprocessing mode 93
 - usage 79
- SQL/DS VM
 - naming programs 95
 - required options 94
- SQLDB option 344
- SQLDBNAM symbolic parameter 132
- SQLID option 345
- SQLPASSWORD option 345
- SQLPKGNM symbolic parameter 133
- SQLPROPT symbolic parameter 133
- SQLSTMDE symbolic parameter 133
- SQLSTOPT symbolic parameter 133
- SQLUSRPW symbolic parameter 133
- SQLVALID option 345
- START subcommand
 - syntax 26, 46, 176, 202
- starting SQL preprocessing mode 93
- static
 - with calllink tag 221
 - with dxfrlink tag 238
- static calls
 - MVS 246, 247
 - VM 250, 251
 - VSE 253, 254
- static links
 - error return codes 256
 - VSE 255
- STOP subcommand
 - syntax 27, 46, 177, 203
- subcommand options 297
- support, contacting IBM for 371
- suppressing messages
 - CICS translator, COBOL compile, and link messages for CICS for OS/2 119
 - Personal Communications messages during file transfer 117
- symbolic parameter
 - assigning values to user-defined 135
 - COB2LIB 131
 - COBLIST 131
 - creating user-defined 134
 - DBDLIB 131
 - description 123
 - DSNLOAD 132
 - DSYS 132
 - ELA 131
 - EZEBLK 130
 - EZECOBOLTYPE 124
 - EZEDATA 124
 - EZEBCS 124
 - EZEDBD 130
 - EZEDD 130
 - EZEDESTLIB 124
 - EZEDESTNAME 124
 - EZEDLBL 130
 - EZEDLI 124
 - EZEDSN 131
 - EZEENTRY 124
 - EZEENV 124
 - EZEGDATE 125
 - EZEGENOUT 125
 - EZEGMBR 125
 - EZEGTIME 125
 - EZELRECL 131
 - EZEMBR 125
 - EZEMBRPATH 125
 - EZEMSG 125
 - EZEPID 125
 - EZEPREPDESTACCOUNT 125
 - EZEPREPDESTDIR 126
 - EZEPREPDESTHOST 126
 - EZEPREPDESTPASSWORD 126
 - EZEPREPDESTUID 126
 - EZEPREPFTPCMDDBCS 126
 - EZEPREPFTPCMDSBCS 126
 - EZEPREPSENDPCMDDBCS 126
 - EZEPREPSESSION 127
 - EZEPREPSP 127
 - EZEPREPSQLDB 127
 - EZEPREPWORKDB 127
 - EZEPSB 127
 - EZEPTYPE 127
 - EZERECFM 131
 - EZESQL 128
 - EZETRAN 129
 - EZETRANSFERTYPE 129
 - EZEUSRID 129
 - EZEVMLOADLIB 129
 - EZEVSELIB 129
 - EZEXAPP 129
- symbolic parameter (*continued*)
 - EZUAUTH 132
 - EZUINST 132
 - file-related 130
 - generation option specifications 214
 - member-related 123
 - predefined 123
 - procedure name 86
 - PROCLIB 132
 - PSBLIB 132
 - PWRCLASS 132
 - RESLIB 132
 - SQLDBNAM 132
 - SQLPKGNM 133
 - SQLPROPT 133
 - SQLSTMDE 133
 - SQLSTOPT 133
 - SQLUSRPW 133
 - user-defined 131
 - values for EFK2MPCB template 120
 - VMDISKADDR 134
 - VMFMODE 134
 - VUSERLIB 134
 - SYMPARM option 346
 - SYNCDXFR option 347
 - SYNCXFER option 347
 - syntax
 - /GROUPNAME 319
 - ASSOCIATE 266
 - GENERATE subcommand for C++ 21
 - PREPARE subcommand for COBOL 175
 - resource association file 266
 - START subcommand 26, 46, 176, 202
 - STOP subcommand 27, 46, 177, 203
 - VALIDATE subcommand for C++ 27, 47
 - VALIDATE subcommand for COBOL 177
 - VALIDATE subcommand for Java 204
 - SYSCODES option 348
 - system name
 - with calllink tag 227
 - SYSTEM parameter 299
- T**
 - table
 - binary 151
 - COBOL program 151
 - conversion 13, 33, 55

- table (*continued*)
 - entries for CICS for OS/2 163
 - table entry, CICS for OS/2 163
 - table templates 67
 - TARGNLS option 348
 - tcPIP
 - with calllink tag 225
 - TEMPAUX, CICS 279
 - template
 - adding a qualifier to a
 - preparation process 113
 - adding a qualifier to a run-time CLIST 111
 - BIND commands 71
 - CICS for VSE/ESA and VSE
 - batch map group preparation templates 66
 - CICS for VSE/ESA and VSE
 - batch preparation templates 64
 - CICS table definition 73
 - CICS tables 72
 - CLIST for MVS/TSO 79
 - default extension 57
 - EFK2CVCL 82
 - EFK2MDLI 81
 - EFK2MEBA 78
 - EFK2META 80
 - EFK2MIMS 81
 - EFK2MPCB 121
 - EFK2MTCL 82
 - EFK2MTDL 81
 - EFK2VDLI 80
 - EFK2VEBA 78
 - examples of modifying 106
 - file and database allocation 80
 - file and database allocation
 - placeholder 81
 - IMS/VS and IMS BMP map
 - group preparation templates 66
 - IMS/VS and IMS BMP
 - preparation 64
 - JCL 75
 - JOB statements 74
 - link-edits 71
 - map group 65
 - modifying 82
 - modifying for MVS
 - environments 84
 - modifying for the OS/400
 - environment 92
 - modifying for VM
 - environments 97
 - modifying for VSE
 - environments 93
 - template (*continued*)
 - MVS and VSE preparation
 - command file 58
 - OS/400 67
 - preparation command file 58
 - preparation file templates for
 - MVS or VSE 62
 - processing 120
 - programs 62
 - reasons to modify 83
 - REXX preparation exec 70
 - run-time files 74
 - run-time JCL for IMS BMP 76
 - run-time JCL for VSE batch 77
 - run-time REXX for VM 78
 - statically linked programs 63
 - table 67
 - types 58
 - used in COBOL generation 57
 - VSE end-of-job 67
 - TEMPLATES option 350
 - TEMPMAIN, file type keyword for
 - CICS environments 280
 - terms
 - default product directory xv
 - template directory xv
 - terms used in this document, list
 - of xii
 - toappl attribute
 - on dxfrlink tag 237
 - TRACE option 350
 - trademarks xi
 - TRANSFERTYPE 351
 - TRANSID option 352
 - TRANSIENT, file type keyword for
 - CICS environments 281
 - TWAOFF option 353
 - types of templates 58
- U**
- understanding multiple levels of
 - generation options files 212
 - UNLOAD option 353
 - URL
 - with calllink tag 230
 - user-defined symbolic
 - parameters 131
 - creating 134
- V**
- valid parameter format and link type
 - for external calls from the test facility 232
 - for GUI programs 231
 - for non-CICS programs 231
 - VALIDATE subcommand 297
 - VALIDATE subcommand 297 (*continued*)
 - example 179
 - example for C++ 28, 47
 - example for Java 204
 - syntax for C++ 27, 47
 - syntax for COBOL 177
 - syntax for Java 204
 - VALIDMIX option 354
 - values for symbolic parameters 120
 - VG
 - with calllink tag 225, 226
 - VisualAge Generator, resource
 - associations part 13, 32, 56
 - VisualAge Generator Developer
 - commands
 - COBOL 167
 - VM 71
 - batch generation example 174
 - file types supported 290
 - generation example 173
 - link-edit templates 71
 - linkage editor control
 - statements 250
 - modifying templates 97
 - preparation REXX exec 143
 - program, with static calls to other
 - programs 250
 - programs, statically called by
 - other programs 251
 - run-time REXX templates 78
 - sample run-time REXX exec 146
 - VMDISKADDR 134
 - VMFMODE
 - user-defined symbolic
 - parameter 134
 - VMLOADLIB option 129, 354
 - VSAM
 - file type 283
 - file type keyword for the CICS
 - for OS/2 environment 283
 - VSE
 - BIND command files not
 - required 264
 - deleting COBOL 114
 - getting ready for
 - preparation 161
 - linkage editor control
 - statements 253
 - list of files transferred 161
 - modifying templates 93
 - modifying the JOB card 95
 - preparation 153
 - preparation file templates 62
 - preparation JCL 142

VSE (*continued*)

- preparation script templates 58
- PREPARE subcommand 176
- program, with static calls to other programs 253
- program and transaction definitions 143
- programs, that are statically called by other programs 254
- programs, with static links to and from other programs 255
- setting COBOL run-time options 95
- SQL preprocessing 93

VSE batch

- DL/I usage 78
- file types supported 292
- generation example 174
- inputs and outputs for COBOL generation 51
- map group preparation templates 66
- preparation templates 64
- run-time JCL templates 77

VSE end-of-job template 67

VSE preparation 161

VSELIB option 355

VUSERLIB symbolic parameter 134

W

Web transaction

- CSOTcpipListener 362
- CSOUiListener 362

Windows NT

- example of generation 26, 45

WORKDB option 356

workstation, deleting COBOL source 113

X

XFER, effect on plans with RT generation option 259

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

VisualAge Generator
Generation Guide
Version 4.5

Publication No. SH23-0263-01

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

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	<input type="checkbox"/>				

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

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	<input type="checkbox"/>				
Complete	<input type="checkbox"/>				
Easy to find	<input type="checkbox"/>				
Easy to understand	<input type="checkbox"/>				
Well organized	<input type="checkbox"/>				
Applicable to your tasks	<input type="checkbox"/>				

Please tell us how we can improve this book:

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

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

Name

Address

Company or Organization

Phone No.



Fold and Tape

Please do not staple

Fold and Tape



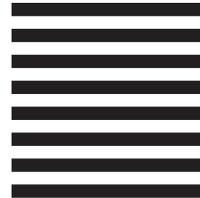
NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

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 062
P.O. Box 12195
Research Triangle Park, NC
27709-2195



Fold and Tape

Please do not staple

Fold and Tape



Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SH23-0263-01

