VisualAge Generator

Client/Server Communications Guide

Version 4.5

SH23-0261-01

Note

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

Third 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.

Contents

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 or both:

AD/Cycle
AIX
AIX/6000
AS/400
CICS
CICS/ESA
CICS OS/2
CICS/MVS
CICS/VSE
CICS/6000
COBOL/370
COBOL/400
C Set + +
DB2
DB2/2
DB2/400
Distributed Relational Database Architecture
DRDA
IBM
IMS
IMS/ESA
Language Environment
MQSeries
MQ
MVS
MVS/ESA
OS/2
OS/390
OS/400
Operating System/2
OpenEdition
PS/2
Presentation Manager
RACF
RISC System/6000
S/390
SAA
SQL/DS
SQL/400

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 for developing client/server systems, and information for developing mixed systems made up of programs developed using VisualAge Generator and programs developed using other tools.

This document contains the following information:
- Distributing program logic using:
  - Calls to remote procedures
  - Message queuing
- Configuring client/server calls
- Accessing distributed data
- Transferring control between VisualAge Generator programs and programs developed using other tools

Who Should Use This Document

This document is intended for program developers, system programmers, and LAN administrators who design and develop programs for a client/server environment.

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]

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.

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]

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

Part 1. Client/Server Concepts

Chapter 1. Introduction to Client/Server Program Development

VisualAge Generator supports the development of programs for client/server environments. These programs can be thought of as having three components:
- User interface
- Logic
- Data

In client/server environments, the user interface and data reside on different computers. The user interface is on a client machine, and the data on a server machine. The machines are connected by a communication link. The program logic can reside either on the client machine, the server machine, or be distributed across machines.

Paradigms for Defining Client/Server Programs

VisualAge Generator supports multiple paradigms for implementing client/server programs. The paradigms follow:

Distributed user interface

User interface is on the client machine. Logic and data are on the server machine.

VisualAge Generator only supports distributed user interface indirectly (and only for character-based user interfaces) through CICS transaction routing.

Distributed data

User interface and logic are on the client machine; at least some program data resides on the server machine.

VisualAge Generator supports access to server data through the following:

- DB2 Distributed Relational Database Architecture (DRDA) for relational databases
- CICS Function Shipping for remote CICS files
- Workstation file servers for workstation files

Distributed logic

User interface is on the client machine and at least some program data is on the server machine. Program logic is distributed between client and server.

VisualAge Generator supports two paradigms for implementing the program-to-program communication needed to distribute logic between client and server. These paradigms follow:

Remote procedure call
The client calls the server passing data to the server and waits for response data to be returned by the server.

Message queueing
Client and server programs communicate by putting messages on queues and retrieving messages from queues. Client and server can run in parallel or asynchronously.

Client/Server Communication

With VisualAge Generator programs, communication between client and server are handled for the program by communication services provided with VisualAge Generator or resource managers.

The communication services you use depends on the client/server paradigm and the runtime environment for your programs. The available common services are the following:

- VisualAge Generator communication support for remote procedure calls from non-CICS clients
- CICS communication support for the following:
  - Remote procedure calls from CICS clients
  - CREATX for starting a remote transaction
  - Function shipping for access to remote CICS files
  - Transaction routing for distributed presentation of character-based user interfaces
- MQSeries communication support for accessing remote message queues
- DB2 communication support for accessing distributed relational databases
- File servers for accessing distributed files

Considerations for Defining Client/Server Programs

When implementing a client/server program using VisualAge Generator, do the following:

- Choose a client/server implementation paradigm suitable for your program and runtime platforms.
- Choose the communication services appropriate to your runtime platforms and paradigm. Understand how to specify the network configuration to the common services (where servers are located in the network and what communication protocols are to be used).

- Understand how to specify to the VisualAge Generator test facility and generator the communication services to be used for client/server functions in generated programs. This information is defined in a generation input file called a linkage table. For information on linkage tables, see "Appendix B. Linkage tables" on page 405.
- Understand how your program will perform the following functions:
  - Control data format conversion

    Workstations and host systems store data in different formats. Character data is represented in ASCII on the workstation and in EBCDIC on the host system. The representation of numeric data also varies. To process data correctly in the receiving system, the program must convert the format of data passed between a host system and a workstation.

    VisualAge Generator performs data conversion, when data is passed between client and server, based on parameter or record data item definition. Variable length records are converted only for the current length of the record, in the number of occurrences item for variably occurring records, or in the record length for variable length items. The conversion table used for data conversion is specified by the CONTABLE attribute in the linkage table.

    Some communication protocols enable the program to control whether conversion is performed at runtime based on the EZECONVT special function word. Other communication protocols perform conversion only when necessary.

    Some VisualAge Generator environments also support the EZECONV service, which can be called to convert data independently of any client/server communication function.
  - Select alternate server locations

    Some communication services enable a program to dynamically select server locations at runtime using the EZELOC special function word; other communication services require a single server location to be specified to the communication services.
  - Handle communication link failures

    Access to distributed data and calls to remote procedures fail if the communication link to the remote system is not available. The client program should be designed to notify the program user that the link was not successful and to provide instructions on how to correct the situation.
  - Handle commit and rollback functions

    The definition of a unit of work and its scope varies with the client/server paradigm, runtime environment, and communication protocol. In some situations, the client controls the unit of work. Database changes made on the server are not committed or rolled back until the client requests a commit or rollback. In other situations, the server unit of work is an independent transaction. Server updates are committed when server processing is complete unless the server program already issued a commit or rollback.

Client/Server Processing Terminology

The following terms are used in describing VisualAge Generator client/server processing:

VisualAge Generator Client Program
A VisualAge Generator program that initiates a request for a service, through remote calls, to a VisualAge Generator called (server) program.

VisualAge Generator Called (Server) Program
A VisualAge Generator program that provides requested services in response to requests, made through remote calls, from VisualAge Generator client programs.

Distributed
A program, function, or data for which the component programs are distributed between two or more interconnected processors. The processors are connected through a communication network.

Location
Location is the logical name of a system where processing occurs.

Local A local program is a server program or data that resides on the same system as the client program that requests a service.

Remote
A remote program is a server program or data that resides on a system other than the system where the client program requests a service.

Part 2. Client or Server Configuration by Platform and Protocol

Chapter 2. Introduction to Client/Server Processing with Synchronous Calls

This chapter contains information that applies to all communication protocols for developing client/server programs. For information on client/server calls using specific protocols, refer to the chapter on configuring your platform.

Figure 1 shows an example of a client/server program using the CALL statement. The client program calls a called batch program that runs on a remote system. When the batch program finishes processing, it returns control to the caller. The call is synchronous; the calling program waits for the remote program to return before continuing.
A call to a server program on a remote system is defined the same way that a

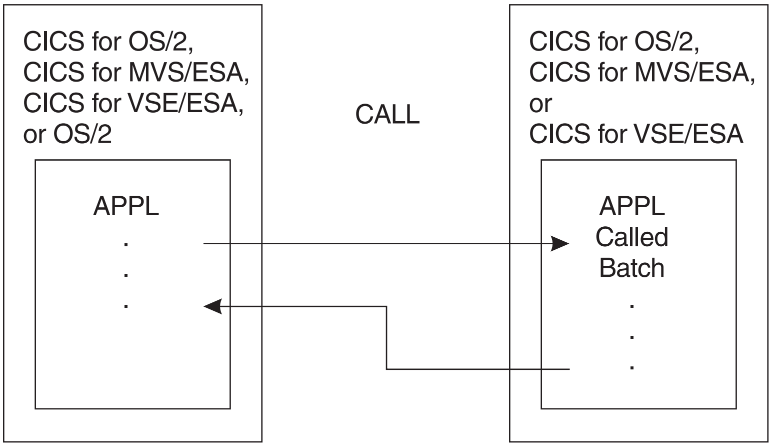


Figure 1. Client/Server Program Using a CALL Statement

call to a local program is defined. The server program is defined like a called batch program; the CALL statement in the client program specifies the arguments that are passed to the server on the call.

At generation, you must specify a CALLLINK tag in the generation linkage table to indicate that the called program is a remote server program. The linkage table also specifies the communication protocol that is used to call the server program. Protocol selection can be bound into the client program at generation, or it can be determined by reading the linkage table again at runtime.

VisualAge Generator supports a variety of client and server system combinations via a number of communication protocol and middleware combinations.

Client Systems

Client programs can be any of the following:
- VisualAge Generator GUI clients on OS/2 or Windows
- VisualAge Generator native C++ programs on OS/2, AIX, or Windows NT, Solaris
- CICS programs
- VisualAge Generator client programs tested using the test facility
- Non-VisualAge Generator client programs that can be programmed to call standard C linkage APIs

Note: Where GUI is specified for OS/2, only GUI clients developed with VisualAge Generator on Smalltalk are supported.

Server Systems

You can generate server programs to run on any of the following environments:
- CICS for MVS/ESA
- IMS
- CICS for VSE/ESA
- OS/400
- OS/2
- CICS for OS/2
- AIX
- CICS for AIX
- Windows NT (C++ and Java)
- CICS for Windows NT
- VM/ESA
- HP-UX
- Solaris
- CICS for Solaris

Supported Middleware

Client programs can call server programs via the following middleware:
- CICS Distributed Program Link (DPL) for CICS server programs
- CICS Client for CICS server programs
- Client Access/400 and Java Toolbox for AS/400 for OS/400 server programs
- APPC/MVS for IMS server programs
- DCE RPC for C++ programs running on OS/2, AIX and Windows NT

- TCP/IP for accessing programs on OS/2, Windows NT, AIX, HP-UX, Solaris, and VM/ESA.

Overview of Supported Environments

Table 2 on page 12 lists the following information for each possible client/server environment:

- Which client environments are supported
- Whether the client can control the unit of work
- Whether the server can control the unit of work
- Whether the server can be a non-VisualAge Generator program

Table 2. VisualAge Generator client/server combinations

Server Platforms	**Client Platforms**					
	OS/2 (GUI, C++, ITF)	**Windows 95 and Windows 98 (GUI)**	**Windows NT and 2000 (GUI, C++, ITF)**	**AIX (C++)**	**CICS**	**Solaris (C++)**
AIX	TCP/IP, DCE	TCP/IP, DCE	TCP/IP, DCE	TCP/IP, DCE, IPC, DIRECT	N/A	TCP/IP, IPC, DIRECT
CICS for AIX	CICS Client	CICS Client	CICS Client	CICS Client	CICS DPL	CICS Client
HP-UX	TCP/IP	TCP/IP	TCP/IP	TCP/IP	N/A	TCP/IP
IMS	APPC/IMS	APPC/IMS	APPC/IMS	N/A	N/A	N/A
CICS for MVS/ESA	CICS Client, LU2	CICS Client	CICS Client	CICS Client	CICS DPL	CICS Client
OS/2	TCP/IP, DCE, IPC, DIRECT	TCP/IP, DCE	TCP/IP, DCE	TCP/IP, DCE	N/A	TCP/IP
CICS for OS/2	CICS Client	CICS Client	CICS Client	CICS Client	CICS DPL	CICS Client
OS/400	CA/400	CA/400,	CA/400, Java400	N/A	N/A	N/A
Solaris	TCP/IP	TCP/IP	TCP/IP	TCP/IP	N/A	TCP/IP, IPC, DIRECT
CICS for Solaris	CICS Client	CICS Client	CICS Client	CICS Client	CICS DPL	CICS Client
VM/ESA	TCP/IP	TCP/IP	TCP/IP	TCP/IP	N/A	TCP/IP
CICS for VSE/ESA	CICS Client	CICS Client	CICS Client	CICS Client	CICS DPL	CICS Client
Windows NT (C++)	TCP/IP, DCE	TCP/IP, DCE	TCP/IP, DCE, IPC, DIRECT	TCP/IP, DCE	N/A	TCP/IP
Windows NT (Java)	N/A	N/A	TCP/IP	N/A	N/A	N/A
CICS for Windows NT	CICS Client	CICS Client	CICS Client	CICS Client	CICS DPL	CICS Client

Notes:

Client controlled unit of work is available for CICS Client, LU2, CA/400, and Java400.

N/A = Not available

APPC = Advanced Program-to-Program Communication.

All server environments support server controlled unit of work.

CICS as a Client Platform refers to the following platforms:

– CICS for AIX,CICS for MVS/ESA, CICS for MVS/ESA, CICS for OS/2, CICS for Windows NT, CICS for VSE/ESA

Defining a Remote Program

The remote program is defined as a called batch program. Define the data to be passed to the called program in the called parameter list. The total number of bytes in the data structures defined for the parameters must be less than or equal to 32567.

Define the logic of the program to reestablish the position in files or databases on each call to the program and to replace or release any records that were read for update before returning to the client program.

Defining the CALL Statement

The syntax of a call to a remote program is the same as a call to a local program.

Specify the data to be passed to the called program using the arguments on the CALL statement. The length of each argument on the CALL statement must be equal to the length of each corresponding parameter in the parameter list specified for the called program.

Arguments that overlap in storage (the same argument is passed more than once or there are multiple definitions of the same record) cannot be passed on a remote call.

Refer to the VisualAge Generator help facility for the correct syntax for the CALL statement.

Testing Client/Server Calls

When you use the VisualAge Generator test facility to test your client/server programs, you have the following options:

- Test both client and server programs from the library.
- Test a client program with a generated server program by including an entry for the server program in the linkage table specified on the **VAGen - Test Linkage** preferences page.
- Test a server program in the library from a generated or packaged client application.

If the client and server programs are in the library and an entry does not exist in the linkage table for the server program, then the following occurs:

- Both programs run under the test facility.

- The logic of both client and server can be traced and monitored in the test facility.
- Data format conversion is not done when the client calls the server, because both programs are running on the same system.
- The contents of the EZELOC and EZECONVT special function words are ignored.
- You must bypass any calls to the EZECONV special function word.

If a client program that is in the library calls a server program and an entry exists in the linkage table for the server program, the test facility reads the linkage table to determine how to call the program. If the server program exists in the library you must select the program and add it to the drop-down list of programs to bypass.

To call a server program that is in the library from a generated or packaged client application, the server program that is to be called must be loaded in the image. The client setup should be performed using the client setup instructions for the communications protocol chosen for the call. The linkage table entries for calling a server in the test facility are as follows:

```
applname=program name
remoteapptype=itf
linktype=remote
       or
linktype=csocall
```

Any remotecomtype that is appropriate from the client system to the target system where VisualAge Generator test facility resides can be used in conjunction with the remoteapptype=itf linkage table entry. A value for the contable attribute is necessary only if required by the client/server operating systems.

If VisualAge Generator test facility is not already running at the time the call is made from the generated or packaged client application, the test facility is started automatically where possible. When the client calls the server program, the test monitor displays an informational message (unless such messages are being suppressed) indicating that the server program is a called batch application and you must press **OK** to continue. At the end of the application the passed data (as modified by the server program), is returned to the client. Unpredictable results might occur if you attempt such a call while multiple sessions of the product are running simultaneously.

Considerations for Testing Java Clients

Java conversion tables are not available for use with the VisualAge Generator Test Facility. It is necessary to create two linkage tables, one to execute your Java client applications inside the Test Facility and one to execute your Java client applications outside of the Test Facility. When executing Java clients in the Test Facility, do not specify a Java conversion table as the value of the CONTABLE attribute in the linkage table. For Java client execution in the Test Facility, specify the conversion tables used for non-Java clients and servers as listed in Table 56 on page 438.

Generating Client/Server Programs

When you generate programs that use client/server processing calls, identify the called program as REMOTE in the linkage table on the linktype attribute, and specify the name of the linkage table when generating both the client and the server programs. The only exception is when you are generating C++ programs; the linkage table entry is only needed for client programs. C++ generated server programs can be called from either a local or remote client.

For example, if a program named CLIENT called a CICS program named SERVER, you would create the following linkage table entry:

```
:calllink applname=server linktype=remote ...
```

Other client/server attributes that are specified on the calllink tag are:
- Program alias (externalname)
- Program library name (library)
- Client or server unit of work control (luwcontrol)
- Communication channel or transaction identifier (serverid)
- Data format conversion control (contable)
- Generation or runtime binding for options (remotebind)
- Parameter format (parmform)
- Protocol selection (remotecomtype)
- Server program type, VisualAge Generator, non-VisualAge Generator, or ITF, where a server application residing in the Repository/ENVY library is executed using VAGen Test Facility (remoteapptype).
- Server location (location)

For more information on the calllink attributes, see “Appendix B. Linkage tables” on page 405.

Identifying the Location of a Server Program

To specify the location of a remote program, use the LOCATION attribute of the CALLLINK tag. The meaning of the location value varies with the communication protocol. See the chapter related to the protocol that you want to use for specific information.

If the same server program resides on multiple server systems, you can code the calling program to load the location into the EZELOC special function word before the CALL. To have the EZELOC value honored by the generated client program, code the location attribute as follows:

```
:calllink  applname=server  linktype=remote  location=EZELOC ...
```

Controlling Data Format Conversion on a CALL

If the program is calling from a client to a server or from a server to a client, and you want data conversion to be performed automatically on the call, specify the contable attribute for the called program in the linkage table, as shown in the following example:

```
:calllink applname=server  linktype=remote  contable=conversion_table_name ...
```

Automatic conversion is performed on the calling system. The program first converts the parameters being sent to the called program from local to remote format, and then converts the values returned from the called program from remote to local format. Conversion is based on the data structure defined for the parameters in the calling program. Use automatic conversion only if the argument values match the structure definition.

To perform conversion using the default conversion table, specify contable=*. The default conversion table performs ASCII to EBCIDIC character conversion, as well as binary numeric conversion. To perform conversion for code pages other than the current code page on your workstation, specify the name of a conversion table that represents the desired code pages. "Appendix C. Converting Between Client and Server Data Formats" on page 433 includes a list of conversion tables provided with the product.

You can control conversion in the following ways:

- To set conversion off on the CALL, specify contable=NONE.

  Note: If the CALL is from a Java GUI and no conversion is required, do not specify the contable attribute.

- To control whether conversion is performed at runtime, specify contable=EZECONVT and code the program to move the conversion table name to EZECONVT before the CALL statement.

  For example, if your program switches server locations using EZELOC, you can set conversion off when the server is on another workstation, and you can set conversion on when the server is on a host system. To set data conversion off, code the program to move blanks to EZECONVT, and then specify EZECONVT as the conversion table on the calllink tag. Move * or a conversion table name to EZECONVT when you want to convert data.

  Note: If the CALL is from a Java GUI do not specify Move *. See Table 55 on page 436 for valid conversion table names to specify on the Move statement.

"Appendix C. Converting Between Client and Server Data Formats" on page 433 contains more information on data format conversion, including a description of the conversion algorithms and a list of conversion tables provided with the product.

Format Conversion for Multi-format Parameters

Do not use automatic conversion if the data structure of the argument can vary from one call to another. Instead, use the EZECONV service routine with redefined records to perform conversion when record formats vary. CALL EZECONV with the redefined record definition that matches the current content of the record buffer.

If a variable length record parameter is passed on a CALL statement and you are using the EZECONV service routine instead of automatic conversion on the CALL, define the record length item or number of occurrences item with the data type of PACK so that the length item has the same format on the host or the workstation. Both the calling and called program depend on the item value being in the correct format for the system where the program is running.

If you are converting data from AIX clients using EZECONV on an MVS or VSE host system, define any binary data items as HEX items in the redefined record to avoid incorrect reversal of the binary items. The host EZECONV routine assumes the binary data is in Intel byte-reversed format.

Format Conversion for Portable Clients and Servers

If the server program can run on a workstation or a host or if the server program must support both workstation and host clients, put the data conversion support in the server program. Have the client pass a parameter on the call indicating the type of system on which the client is running. Have the server compare the client system to the host system and perform conversion as required using EZECONV. Encode the system type in a HEX data item because HEX item values remain unchanged during format conversion.

Calling Server Programs from 4GL Java Clients

When defining the linkage table for a generated 4GL Java client, the LINKTYPE and CONTABLE attributes of the CALLLINK tag must be configured for Java clients. There are also special considerations for enabling Java applets to call server programs.

CSOCALL is the only valid value for the LINKTYPE attribute. All calls to server programs made by 4GL Java clients are executed through the Common Services option for VisualAge Generator.

Valid values for the CONTABLE attribute are of the form CSOBXXXX. The first three characters are a prefix. The ′B′ character is for the byte order of the target system. Valid values for the ′B′ character are ′X′ for Unix systems (AIX, HP-UX, Solaris, etc.), ′I′ for Intel systems, and ′E′ for EBCDIC systems. The ″XXXX″ string is for the code page of the target system. For more information on conversion tables for Java clients, see Table 55 on page 436.

Calling Server Programs from Java Applets

Java security restrictions do not allow Applets loaded off of remote machines to call server programs from the machine on which they are executing. To enable Java Applets to call server programs, an RMI server called the SessionManager has been provided to act as an interface to the Common Services option. For information on configuring the SessionManager to enable applets to call server applications, see "Chapter 18. VisualAge Generator JavaBeans Wrappers and Enterprise Beans" on page 289.

For a generated 4GL Java Client applet to be able to contact the SessionManager it must know the location of the SessionManager. Modify the vgj.properties file on your web server so that the vgj.powerserver.location property is set to the hostname of your webserver. For example, if your applet is served to the browser from http://myserver.com/applets/myapplet, then the vgj.powerserver.location property in vgj.properties should read:

```
vgj.powerserver.location=myserver.com
```

Note: If you created your own properties file or had one generated when you generated the program, this property might be included in that file.

The SessionManager must be located on the machine that the applet was downloaded from.

The vgj.properties file can be found in the Common Services install directory. For AS/400, the vgj.properties file is located in /QVGEN/LIB if you have VisualAge Generator Server for AS/400 installed.

Calls between Server Programs and VAGen Java Server Programs

In addition to clients, VAGen Java server programs can receive and make calls to other server programs. They can make and receive local calls from other VAGen Java server programs running on the same machine. They can also receive remote calls from other Java and non-Java programs, make calls to other remote servers and start local and remote Web transactions. For specific examples of these types of calls, see the appropriate protocol section in "Configuring a Windows NT Server" on page 264.

Calls to server programs are specified in linkage tables either at generation time or at runtime. Generation-time binds of remote calls must be fully defined. If remotebind=RUNTIME is specified in the linkage table at generation time, and you elect to have a properties file generated, default properties will be set for you. (See “Configuring a Windows NT Server” on page 264 for examples.)

Note: Generation of VAGen Java server programs is currently only supported for the Windows NT environment.

Local Calls to VAGen Java Server Programs

A local call is one in which one Java program runs another inside the same Java virtual machine (JVM). This is only possible between VAGen Java servers running on the same machine. To make a local call, the linktype attribute must be DYNAMIC, STATIC, CSOCALL, or unspecified.

If CSOCALL is used, the remotecomtype must be DIRECT and the remoteapptype must be VGJAVA. In all cases the package attribute must specify the called server’s package. If the package isn’t specified, the default is the package of the current program, but only if the linkage options were specified at generation. Resources, such as shared tables, database connections, and Common Services resources, are shared between programs unless the DIRECT protocol is used.

Conversion tables are ignored unless a Bidi conversion table is used with the DIRECT protocol. Local calls can be made from Java GUIs and wrappers if the linktype is CSOCALL, the remotecomtype is DIRECT, and the remoteapptype is VGJAVA. The GUI or wrapper must have access to vgjwgs.jar.

For a specific example on how to specify linkage for local calls between VAGen Java servers, see “Sample linkage table entries” on page 430.

Remote Calls from Java Server Programs

A remote call is either a call between a Java and a non-Java server program or two Java server programs running in different JVMs. The calling program and the called program may or may not be on the same machine.

Java servers can use any middleware supported by the Common Services on Windows NT. When a remote call is made with remotecomtype=TCPIP, the serverid is required and it must be a port number. When remotebind=RUNTIME is specified in the generation linkage table, linkage tables are to be loaded at run time. If the linkage table is named LINK, the program will look for a property named cso.linkagetable.LINK in its properties file. If the property is defined, its value is the name of another properties file, which contains the linkage information to be used. If the property is not defined, the linkage information may be contained in the program's properties file.

For a specific example of this type of calls, see the appropriate protocol section in "Configuring a Windows NT Server" on page 264.

Remote Calls to Java Server Programs

Remote VG Java servers can only be called using TCP/IP. The remoteapptype should be VGJAVA. Only Bidi conversion tables are supported. TCPIP calls to Java programs can use the package attribute from the linkage table. The package name is passed along with the rest of the data on the call. If the package is not specified and remotebind is GENERATION, the package of the current program is used by default. remoteapptype=VGJAVA is not supported in Smalltalk GUIs, C++ programs, or COBOL programs.

For a specific example of this type of calls, see the appropriate protocol section in "Configuring a Windows NT Server" on page 264.

Calling Java Web Transactions

Like remote Java servers, Java Web transactions that are not on the same machine as the gateway servlet can be run using the TCPIP protocol with remoteapptype=VGJAVA. A conversion table is required when a Web transaction is contacted using TCPIP. The character representing the byte order should be 'J', as in CSOJ1252. If a Java Web transaction is on the same machine as the GatewayServlet, it may be called locally. In this case, the DIRECT protocol can be used with remoteapptype=VGJAVA. javaProperty, commtype, and remoteapptype are the only properties required for a DIRECT call. Only Bidi data conversion tables are supported. The DIRECT protocol is only available when the GatewayServlet is running on Windows NT.

For a specific examples of this type of calls, see the appropriate protocol section in "Configuring a Windows NT Server" on page 264.

Committing Changes for Remote Called Programs

Use the luwcontrol attribute on the calllink tag to specify whether the client or server program controls the unit of work.

With server-controlled unit of work, each server call is a separate unit of work independent of the client's unit of work. Commit (or roll back on abnormal termination) occurs automatically on return from the server program.

With client-controlled unit of work, server updates are not committed or rolled back until the client program requests a commit or rollback. Include EZECOMIT and EZEROLLB calls in the client program to request commits and rollbacks.

Use server-controlled server unit of work whenever possible because it provides the best performance and least serialization from locked resources.

If client-controlled unit of work is used, you should commit or roll back at the earliest possible point to avoid lockout. At a minimum, always code the program to commit or roll back before returning control to the user interface to wait for the next user-initiated action. This ensures that data locks are not held while the user is away from the workstation.

See Table 2 on page 12 for information on the environments that support client or server controlled units of work.

User Authentication

The user of the client program must be identified and authenticated through password checking. The authentication process varies with the communication middleware being used and is described in the middleware-specific chapters.

When the requirement for a userid and password is detected, common services will invoke a userid/password prompting DLL. The default prompt for OS/2, Windows, AIX, and HP-UX is CSOUIDPW. CSOUIDPW reads the environment variables CSOUID and CSOPWD to obtain the userid and password. You can use the CSOPRMPT prompt DLL instead of the default CSOUIDPW by setting the environment variable CSOUEXIT to CSOPRMPT (that is, set CSOUEXIT=CSOPRMPT). CSOPRMPT is a GUI prompt panel.

Alternative prompting mechanisms can be written and used by setting the CSOUEXIT environment variable to the name of the alternative prompting DLL. For example, see the CSOPRMPT source code in the sample directory.

If you use the Java400 protocol to access an AS/400 server program, you must program the Java GUI application to supply a user ID and password for user authentication.

Accessing DL/I Databases in Remote Called Programs

Server programs running on IMS, CICS for MVS/ESA, or CICS for VSE/ESA systems can access DL/I databases. The PSB for a server program remains scheduled until the unit of work ends.

If you are generating the server program for IMS, always include EZEDLPSB in the called parameter list for the server program and in the CALL statement in the client program.

When generating for CICS systems, the following restrictions apply if more than one call to a DL/I server call is issued in the same unit of work:
- All calls must go to the same location.
- EZEDLPSB must be included in the called parameter list and specified as an argument on the call.

Communication Error Handling

If an error occurs on a call to a server program, the program behaves differently depending on the type of error and whether the REPLY option was coded on the CALL statement, as shown in Table 3:

Table 3. Error Handling for Remote Calls

Type of Error	REPLY Option Specified	EZERT8 Setting	Log or Trace Message on Client System	Log or Trace Message on Server System	Program Ends With Error Message to User
No error, successful completion	No	Unchanged	No	No	No
No error, successful completion	Yes	0	No	No	No
Communication failure	Yes	Middleware dependent error code	Yes	No	No
Communication failure	No	Unchanged	Yes	No	Yes
Terminating error in server	Yes	Error code	Yes	Yes for CICS	No
Terminating error in server	No	Unchanged	Yes	Yes for CICS	Yes

You can obtain trace information for remote server calls from non-CICS client programs by setting the CSOTROPT and CSOTROUT environment variables.

CSOTROUT
Specifies the trace file name. The default name is csotrace.out in your current directory.

CSOTROPT
Specifies the level of trace information collected.

CSOTROPT=1
Only errors are collected in the trace file. This is the default if CSOTROPT is not set.

CSOTROPT=2
All traces are collected in the trace file

You can collect additional problem information for calls made using VisualAge Generator client/server common services. Refer to the *VisualAge Generator Messages and Problem Determination Guide* for additional information.

For CICS client programs, all error messages are written to the error message transient data queue associated with the transaction.

Communication Failures

Communication failures are detected by the communication middleware and can be caused by set-up failures or communication link problems. You can handle errors on server calls in the program by specifying the REPLY option on the CALL statement and testing the middleware-dependent return code in EZERT8 following the call. This error handling should include issuing a rollback for any updates that were made. If the REPLY option is omitted, the client program ends with an error message on a communication failure, and a rollback is automatically issued. In either case, the client program writes a message describing the error to the log or trace file that is associated with the client program.

If server-controlled unit of work is in effect (server commits on return), a successful return indicates that the desired database or file updates have been successfully made on the server system. If client-controlled unit of work is in effect, a successful return from an EZECOMIT call indicates that server updates were completed.

An unsuccessful return usually, but not always, implies that the updates were not committed. Design client programs to validate whether a previously requested update has completed successfully on a communication or commit failure before attempting to do the update again.

Errors in Generated Server Programs

If VisualAge Generator Server, VisualAge Generator Server for AS/400, or VisualAge Generator Server for MVS, VSE, and VM detects an error that prevents a server program from completing, the messages describing the error are logged on the server system according to the diagnostic control options in effect on the server system. The server program returns an error code in the communication buffer. The client program ends with a rollback request, a message indicating the server program did not successfully complete, and the time and date at which the error occurred.

The error return area is generated as an additional parameter for remote server programs that are generated as CICS programs. The format of the return area is shown in the following list. All fields are packed decimal and have the same format on the host and workstation:

Bytes	Description
1	Reserved–should always be binary zero
2–4	Return code
5–8	Error date (00YYMMDD)
9–12	Error time (00HHMMSS)

The client program sets the return area to binary zeros. The server does not modify the area unless an error is detected that prevents the server program from continuing. If that occurs, the server program writes messages describing the error to the error log and stores a return code of 693, the error date, and the error time stamp in the return area. The error date and time reported in the message can be used to find the related messages in the server error log.

Single Server for Local and Remote Client Programs

Single CICS Server

If you want to generate a server program that accepts calls from client programs running on either a local system or a remote system, generate the server program, the local clients, and the remote clients with linktype=REMOTE in the linkage table. Although the called program might reside on the same system as the calling program, the client program uses the remote call interface.

Single Servers for OS/2, AIX, OS/400, and IMS

If a non-CICS server program is generated as a remote program, the generated program will accept calls from clients on both the local operating system and on remote systems.

To use a local operating system call for a local client, generate the local client with a linkage table that specifies linktype=DYNAMIC and parmform=OSLINK instead of linktype=REMOTE. Continue to specify linktype=REMOTE when you generate the server program or when you generate a remote client program.

Calling Non-VisualAge Generator Server Programs

The server program can be either a generated program or a non-VisualAge Generator program developed with another tool. Specify remoteapptype=NONVG on the calllink tag if the server program was not generated using VisualAge Generator. Ensure that the definition of the parameters in the server program matches the definition of the arguments specified on the CALL statement. Ensure that the parameter format matches the format specified on the parmform attribute on the calllink tag. Do not include the extra error return parameter in the parameters defined in the server program.

Chapter 3. AIX Platform

How to use this chapter

This chapter contains platform and protocol information necessary to configure a client program or a server program. The chapter is divided into configuration sections: client programs and server programs. Each configuration section is organized by protocol. Each protocol section contains information organized by target platforms.

Use the following method to locate the information you need:

1. Verify that this chapter is appropriate to your runtime environment.

   Ensure that the runtime environment of your program matches the platform on which this chapter is based. Turn to the appropriate chapter if the runtime environment of your program does not match the platform named in the title of this chapter.
2. Determine which type of program you want to configure: client program or server program.
3. Determine which configuration section in the chapter is appropriate for your program.
4. Find the intended target platform of the program you are configuring in the Summary Table of Valid Clients and Protocols or Summary Table of Valid Servers and Protocols.

   If you are configuring a client program, find the target platform of the intended server program in the column of valid platforms.

   If you are configuring a server program, find the target platform of the intended client program in the row of valid platforms.
5. Chose a protocol based on the information in the table.

   If you are configuring a client program, find the cell in the table where the row of the intended target platform intersects the column of the client platform. Select a protocol from the list of available protocols for this client/server combination.

   If you are configuring a client program, find the cell in the table where the column of the intended target platform intersects the row of the server platform. Select a protocol from the list of available protocols for this client/server combination.
6. Determine the name of the appropriate protocol section within the configuration section.

Turn to the part of the configuration section indicated by the protocol name. Here you will find information common to all target platforms valid for this protocol.

7. Determine the name of the appropriate target platform section within the protocol section.

   You will find platform specific information for the chosen protocol in the section named for your intended target platform.

Example

```
Chapter: AIX Platform
Type of program to configure: Client program
Configuration section: Configuring an AIX client
Intended target platform: CICS for MVS/ESA
(i.e. a CICS for MVS/ESA server program)
Chosen protocol: CICS Client
Protocol section: CICS Client Protocol
Target platform section: Configuring an AIX Client for a
CICS for MVS/ESA Server
```

You will encounter syntax diagrams in this chapter. The diagrams are used to show programming syntax as follows:

```
►►─:RequiredTerm──RequiredTerm=RequiredValue──RequiredTerm=─┬─OptionalValue─┬──────────►
                                                            └─OptionalValue─┘

►─┬─────────────────────────────────────────┬─┬──────────────────────────────┬──────────►
  │                 ┌─DefaultValue──┐        │ └─OptionalTerm=RequiredValue─┘
  └─OptionalTerm=───┴─OptionalValue─┴───────┘

►─┬─────────────────────────────────────────┬──────────────────────────────────────────►◄
  └─OptionalTerm=─┬─OptionalValue─┬─────────┘
                  └─OptionalValue─┘
```

RequiredTerm	A required term is a term that must be specified. Required terms are depicted in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required terms.
RequiredValue	A required value is a value that must be specified for a term. A required value is shown in italics in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required values.
OptionalTerm	An optional term is a term that can be

	specified. An optional term is depicted in syntax diagrams in an alternate path from the preceding term.
OptionalValue	An optional value is a value that can be specified for a term. An optional value is shown in italics in syntax diagrams in an alternate path from the preceding term along with other optional values or a default value.
DefaultValue	A default value is the value that is specified by default for a term. A default value is shown in italics in syntax diagrams in an alternate path above other optional values.

Configuring an AIX Client

Summary Table of Valid Servers and Protocols

Table 4. Valid Servers and Protocols for VisualAge Generator Clients on the AIX Platform

Server Platforms	Protocols for AIX Client Platform
AIX	TCP/IP, DCE, IPC, DIRECT
CICS for AIX	CICS Client
HP-UX	TCP/IP
CICS for MVS/ESA	CICS Client
OS/2	TCP/IP, DCE
CICS for OS/2	CICS Client
Solaris	TCP/IP
CICS for Solaris	CICS Client
VM/ESA	TCP/IP
CICS for VSE/ESA	CICS Client
Windows NT (C++)	TCP/IP, DCE
Windows NT (Java)	NA
CICS for Windows NT	CICS Client

CICS Client Protocol

User Authentication

The user authentication exit (see “User Authentication” on page 21) provides the user ID and the password specified on the ECI call. The program user must be authorized to run the transaction associated with the server call.

The user exit can return NULLs for the userid and password. The default exit returns the contents of the environment variables CSOUID and CSOPWD as the userid and password. If nulls are specified on the ECI call, the CICS Client determines the user ID and password in a system-dependent fashion. Refer to the CICS Client documentation for your environment for further information.

Setting Up Communication Links

The communication link between the client and server systems must be defined to the CICS server system and client products to allow an ECI call to flow from a CICS Client product to server systems. CICS Client, the communications software, is installed and configured on the client. Refer to CICS CLIENT documentation. The CICS server environment must have a "listener" defined and requires other entries if remote programs are called. Refer to CICS documentation for more information. Refer to the CICS intercommunication documentation for your CICS systems and client products for additional information.

Identifying the CICS Transaction for the Server

The CICS transaction name associated with the server program is specified in the `serverid` linkage table attribute. If not specified, the default transaction is the CICS-supplied mirror transaction, CPMI.

Controlling the Unit of Work

Extended Units of Work: Multiple synchronous calls to CICS servers can be issued from the same client. You can use the extended unit of work feature of ECI to include several calls to the same system within the same unit of work by specifying `luwcontrol=CLIENT` in the linkage table for the server programs. For CICS servers, the default value for LUWCONTROL is client unit of work. The extended unit of work ends when the client program calls EZECOMIT or EZEROLLB, which results in an ECI call to commit or roll back any extended transactions that are currently active.

A separate ECI extended unit of work (CICS transaction) is started for each unique `serverid/location` pair. Servers on the same system running under the same SERVERID (transaction name) are part of the same CICS extended transaction. A client EZECOMIT or ROLLBACK call ends all the extended transactions currently in effect for the client.

The server cannot issue EZECOMIT or EZEROLLB calls if the client unit of work was in effect for the server call.

Server Unit of Work: You can specify `luwcontrol=SERVER` in the linkage table. With this specification, each server call is a separate unit of work. The server program can issue commit or rollback requests as well.

Data Format Conversion

Code format conversion is performed on the server call as specified on the `contable` linkage table attribute and EZECONVT special function word. Code is converted based on the structure of the arguments specified on the CALL statement in the client program.

Error Handling

The standard error handling procedures described in “Communication Error Handling” on page 22 are supported with CICS ECI. EZERT8 is set to the decimal value of the client access service reason code. The reason code is the same as the number portion of error message CSOnnnna as listed in the *VisualAge Generator Messages and Problem Determination Guide* document.

Specifying Parameter Format

Parameter format (the `parmform` attribute on the calllink tag) can be specified as COMMDATA (data values passed in the COMMAREA) or COMMPTR (pointers passed in the COMMAREA).

Using COMMPTR Format: Use COMMPTR for calling COBOL subroutines that were developed to use the non-VAGen parameter passing convention that uses pointers in the CICS COMMAREA. When the pointer format is used, the generated client program calls an intermediate catcher program (ELACSV, provided with VisualAge Generator host services) via the ECI instead of calling the server program directly. The catcher program receives the parameter data from the client, builds a pointer list in the COMMAREA that points to the individual parameters, and links to the server program. A program processing table (PPT) must be defined for the ELACSV program on the CICS server system.

COMMPTR format is only supported for MVS CICS, CICS for VSE/ESA, and CICS OS/2 servers.

Using COMMDATA Format: Use COMMDATA for calling generated programs and for calling COBOL subroutines where possible. COMMDATA provides better performance in calling remote programs. With COMMDATA, the server program is called directly without going through the catcher program.

Configuring an AIX Client for a CICS for OS/2 Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►—:calllink—applname=program name—linktype=–REMOTE—remotecomtype=–CICSCLIENT——————►
```

```
►─┬──────────────────────────────────────┬──┬──────────────────────────────┬──────────►
  └─contable=─┬─conversion table name─┬──┘  └─location=─┬─EZELOC──────┬─┘
              ├─'*' ──────────────────┤                 └─system name─┘
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬─────────────────────────────┬─┬─────────────────────────────┬──────────────────────►
  │             ┌─CLIENT─┐      │ └─parmform=─┬─OSLINK─────┬─┘
  └─luwcontrol=─┴─SERVER─┴──────┘             ├─COMMPTR────┤
                                              ├─COMMDATA───┤
                                              └─CICSOSLINK─┘

►─┬──────────────────────────────┬─┬──────────────────────────────────┬──────────────►
  └─remoteapptype=─┬─VG────┬─────┘ │              ┌─GENERATION─┐      │
                   ├─NONVG─┤       └─remotebind=─┴─RUNTIME────┴──────┘
                   └─ITF───┘

►─┬───────────────────────────────────┬───────────────────────────────────────────►◄
  └─serverid=──server identifier──────┘
```

The following attributes are ignored:
- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST contable=BINARY
```

Configuring an AIX Client for a CICS for Windows NT Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─CICSCLIENT──────►

►─┬──────────────────────────────────────┬──┬──────────────────────────────┬──────────►
  └─contable=─┬─conversion table name─┬──┘  └─location=─┬─EZELOC──────┬─┘
              ├─'*' ──────────────────┤                 └─system name─┘
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘
```

```
►─┬──────────────────────────────┬─┬────────────────────────────────┬──────►
  │               ┌─CLIENT─┐     │ └─parmform=─┬─OSLINK─────┬─┘
  └─luwcontrol=───┴─SERVER─┴─────┘             ├─COMMDATA───┤
                                               └─CICSOSLINK─┘
```

```
►─┬───────────────────────────────┬─┬───────────────────────────────────┬─►
  └─remoteapptype=─┬─VG──────┬────┘ │              ┌─GENERATION─┐       │
                   ├─NONVG───┤      └─remotebind=──┴─RUNTIME────┴───────┘
                   └─ITF─────┘
```

```
►─┬─────────────────────────────────┬────────────────────────────────────►◄
  └─serverid=───server identifier───┘
```

The following attributes are ignored:
- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST contable=BINARY
```

Configuring an AIX Client for a CICS for AIX Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─CICSCLIENT─────►
```

```
►─┬─────────────────────────────────────────┬─┬──────────────────────────────┬──────►
  └─contable=─┬─conversion table name─┬─────┘ └─location=─┬─EZELOC──────┬───┘
              ├─'*'───────────────────┤                    └─system name─┘
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘
```

```
►─┬──────────────────────────────┬─┬────────────────────────────────┬──────►
  │               ┌─CLIENT─┐     │ └─parmform=─┬─OSLINK─────┬─┘
  └─luwcontrol=───┴─SERVER─┴─────┘             ├─COMMDATA───┤
                                               └─CICSOSLINK─┘
```

```
►─┬──────────────────────────────┬─┬─────────────────────────────────┬─────────────────►
  └─remoteapptype=─┬─VG─────┬────┘ │                ┌─GENERATION─┐   │
                   ├─NONVG──┤      └─remotebind=────┴─RUNTIME────┴───┘
                   └─ITF────┘

►─┬─────────────────────────────────────┬──────────────────────────────────────────────►◄
  └─serverid=──server identifier────────┘
```

The following attributes are ignored:
- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST contable=BINARY
```

Configuring an AIX Client for a CICS for MVS/ESA Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─CICSCLIENT──────►

►─┬──────────────────────────────────────────┬─┬─────────────────────────────────┬─────►
  └─contable=─┬─conversion table name──────┬─┘ └─location=─┬─EZELOC──────────┬──┘
              ├─'*' ───────────────────────┤               └─system name─────┘
              ├─EZECONVT───────────────────┤
              ├─NONE───────────────────────┤
              └─BINARY─────────────────────┘

►─┬──────────────────────────────┬─┬─────────────────────────────────┬─────────────────►
  │                ┌─CLIENT─┐    │ └─parmform=─┬─OSLINK─────────┬────┘
  └─luwcontrol=────┴─SERVER─┴────┘             ├─COMMPTR────────┤
                                               ├─COMMDATA───────┤
                                               └─CICSOSLINK─────┘

►─┬──────────────────────────────┬─┬─────────────────────────────────┬─────────────────►
  └─remoteapptype=─┬─VG─────┬────┘ │                ┌─GENERATION─┐   │
                   ├─NONVG──┤      └─remotebind=────┴─RUNTIME────┴───┘
                   └─ITF────┘

►─┬─────────────────────────────────────┬──────────────────────────────────────────────►◄
  └─serverid=──server identifier────────┘
```

The following attributes are ignored:

- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST contable=ELACNxxx
```

Note: Specifying the CONTABLE attribute requires the three character NLS code appropriate to your language. See “Conversion Table by Language and Platform” on page 436 for more on information on specifying the CONTABLE attribute.

Configuring an AIX Client for a VSECICS Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►—:calllink—applname=program name—linktype=—REMOTE—remotecomtype=—CICSCLIENT————►

►—┬————————————————————————————————————————┬—┬———————————————————————————┬———————►
  └contable=—┬conversion table name┬———————┘ └location=—┬EZELOC—————┬———┘
             ├'*'——————————————————┤                    └system name┘
             ├EZECONVT—————————————┤
             ├NONE—————————————————┤
             └BINARY———————————————┘

►—┬—————————————————————————————┬—┬——————————————————————————┬—————————————————►
  │                 ┌CLIENT┐    │ └parmform=—┬OSLINK————┬———┘
  └luwcontrol=——————┴SERVER┴————┘            ├COMMPTR———┤
                                             ├COMMDATA——┤
                                             └CICSOSLINK┘

►—┬——————————————————————————————┬—┬———————————————————————————————————┬————————►
  └remoteapptype=—┬VG————┬———————┘ │                ┌GENERATION┐       │
                  ├NONVG—┤         └remotebind=—————┴RUNTIME———┴———————┘
                  └ITF———┘

►—┬—————————————————————————————————————┬——————————————————————————————————————►◄
  └serverid=——server identifier—————————┘
```

The following attributes are ignored:
- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST contable=ELACNxxx
```

Note: Specifying the CONTABLE attribute requires the three character NLS code appropriate to your language. See “Conversion Table by Language and Platform” on page 436 for more on information on specifying the CONTABLE attribute.

Configuring an AIX Client for a CICS for Solaris Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►─:calllink─applname=program name─linktype=─REMOTE─remotecomtype=─CICSCLIENT──────►

►─┬──────────────────────────────────────┬─┬──────────────────────────────┬────────►
  └─contable=─┬─conversion table name─┬─┘ └─location=─┬─EZELOC──────┬─┘
              ├─'*' ──────────────────┤               └─system name─┘
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬────────────────────────────┬─┬──────────────────────────┬──────────────────────►
  │             ┌─CLIENT─┐     │ └─parmform=─┬─OSLINK────┬─┘
  └─luwcontrol=─┴─SERVER─┴─────┘             ├─COMMDATA──┤
                                             └─CICSOSLINK┘

►─┬──────────────────────────────┬─┬──────────────────────────────┬────────────────►
  └─remoteapptype=─┬─VG────┬─────┘ │             ┌─GENERATION─┐   │
                   ├─NONVG─┤       └─remotebind=─┴─RUNTIME────┴───┘
                   └─ITF───┘

►─┬───────────────────────────────┬───────────────────────────────────────────────►◄
  └─serverid=──server identifier──┘
```

The following attributes are ignored:
- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST contable=BINARY
```

DCE Protocol

Processing Flow for VisualAge Generator DCE Common Services Remote Call

This section shows the processing flow for a VisualAge Generator DCE common services remote call.

1. VisualAge Generator DCE Server is started using a configuration file which specifies the DCE principal name that the server will obtain its DCE authorizations from (equivalent to a DCE userid), the location and the serverid name for binding information advertising, the Access Control List (ACL) object for client authorization, and the server programs that the server is authorized to process. The server programs are specified in one of two different groups; those in which secure DCE (authenticated RPC) communications are required and those which can be accessed via unsecured (unauthenticated RPC) DCE communications.

   The server obtains an object UUID for each program from the CDS object /.:/Servers/VAGenerator/SERVERID/program. If the program object is not defined, one will be created for it. The server uses the program object UUIDs when it advertises its binding information.
2. VisualAge Generator DCE Server advertises each of the programs that it services. The Cell Directory Services (CDS) location used for advertising the binding information is /.:/Servers/VAGenerator/SERVERID/LOCATION.
3. VisualAge Generator client retrieves the object UUID for the program from /.:/Servers/VAGenerator/SERVERID/program-name. It then uses the object UUID to request the binding information for a server which services the program from the CDS location /.:/Servers/VAGenerator/SERVERID/LOCATION. If there are multiple server bindings that match the search criteria, DCE will randomly return one of them.
4. VisualAge Generator client will setup for authenticated RPC, if DCESECURE is specified in the linkage table.
5. VisualAge Generator client performs data conversion on passed parameters as specified in the `contable` attribute of the client linkage table (runtime or generation time as applicable).
6. VisualAge Generator client makes remote call to VisualAge Generator DCE Server.
7. VisualAge Generator DCE Server checks if VisualAge Generator client is authorized to use server program (via DCE CDS Access Control List) and if the appropriate level of communication security is being used from the client.
8. VisualAge Generator DCE Server checks if the server program requested is one that it is authorized to process (via initial configuration file).

9. VisualAge Generator DCE Server processes client request, closes Logical Unit of Work, and returns data to client.

User Authentication and Authorization

User authentication is performed on the client via the DCE security server using the client's DCE login identifier. The VisualAge Generator DCE server checks whether the client is authorized to call the DCE server using DCE ACL security services. User authorization is performed via the DCE ACL security services. The VisualAge Generator DCE server is told at startup time the DCE object ACL to use for checking client authorization for running the called server programs. There is only one ACL used per VisualAge Generator DCE server; therefore, the authorization is at the VisualAge Generator DCE server level and not the called program level. If a called program requires a special ACL, then another VisualAge Generator DCE server will have to be created or started. The test ACL attribute determines whether or not the client is authorized to execute the server program (if the client has test privileges on the ACL object, then the client is authorized to execute all server programs provided by the server).

Controlling the Unit of Work

All calls via the DCE common services are server units of work. All resource changes are committed when the server returns to the calling program.

Data Format Conversion

Code format conversion is performed on the server call as specified on the `contable` attribute for the server entry in the linkage table. Data is converted based on the structure of the parameters specified on the server call in the client program.

Error Handling

The standard error handling procedures described in "Communication Error Handling" on page 22 are supported with VisualAge Generator DCE common services. EZERT8 is set to the decimal value of the client access service reason code. EZERT8 is only set when the REPLY option is coded on the call to the remote server program. If a visual link is used to make the call from a GUI program, the REPLY option is used on the call.

Any errors trapped by DCE are passed to the client program with a corresponding CSO error message. The error message contains an insert with the DCE mnemonic. A non-zero return code from the called program is passed back to the client program with a corresponding CSO error message. The error message contains an insert with the return code from the called program. VisualAge Generator return codes are documented in the help for the message.

All errors are traced to the CSO trace file on the client and server machines, as applicable.

Configuring an AIX Client for an OS/2 Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call server programs via the DCE common services:

```
►►──:calllink──applname=program name──linktype=─REMOTE──────────────────────────►

►─remotecomtype=──┬─DCE───────┬──┬─────────────────────┬────────────────────────►
                  └─DCESECURE─┘  └─parmform=─COMMDATA──┘

►─┬──────────────────────────────────────┬─┬─────────────────────────────┬──────►
  └─contable=──┬─conversion table name─┬─┘ └─location=──┬─EZELOC──────┬─┘
               ├─'*'───────────────────┤                └─system name─┘
               ├─EZECONVT──────────────┤
               ├─NONE──────────────────┤
               └─BINARY────────────────┘

►─┬──────────────────────────────────┬─┬──────────────────────────────────┬─────►◄
  │             ┌─GENERATION─┐       │ └─serverid=───server identifier───┘
  └─remotebind=─┴─RUNTIME────┴───────┘
```

The following attributes are ignored:
- remoteapptype
- externalname
- library
- luwcontrol

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=DCE
   location=Server1 contable=BINARY remotebind=RUNTIME serverid=Test
```

Configuring an AIX Client for a Windows NT Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call server programs via the DCE common services:

```
►►──:calllink──applname=program name──linktype=─REMOTE──────────────────────────►

►─remotecomtype=──┬─DCE───────┬──┬─────────────────────┬────────────────────────►
                  └─DCESECURE─┘  └─parmform=─COMMDATA──┘
```

```
►──┬──────────────────────────────────────┬──┬──────────────────────────────┬──────────────►
   └─contable=──┬─conversion table name─┬─┘  └─location=──┬─EZELOC──────┬─┘
                ├─'*' ──────────────────┤                 └─system name─┘
                ├─EZECONVT──────────────┤
                ├─NONE──────────────────┤
                └─BINARY────────────────┘

►──┬──────────────────────────────┬──┬─────────────────────────────────────┬──────────────►◄
   │                ┌─GENERATION─┐ │  └─serverid=───server identifier─────┘
   └─remotebind=──┴─RUNTIME────┴─┘
```

The following attributes are ignored:
- remoteapptype
- externalname
- library
- luwcontrol

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=DCE
   location=Server1 contable=ELAWIxxx remotebind=RUNTIME serverid=Test
```

Note: Specifying the CONTABLE attribute requires the three character NLS code appropriate to your language. See "Conversion Table by Language and Platform" on page 436 for more on information on specifying the CONTABLE attribute.

Configuring an AIX Client for an AIX Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call server programs via the DCE common services:

```
►►──:calllink──applname=program name──linktype=─REMOTE──────────────────────────────────────►

►─remotecomtype=──┬─DCE───────┬──┬──────────────────────┬──────────────────────────────────►
                  └─DCESECURE─┘  └─parmform=─COMMDATA───┘

►──┬──────────────────────────────────────┬──┬──────────────────────────────┬──────────────►
   └─contable=──┬─conversion table name─┬─┘  └─location=──┬─EZELOC──────┬─┘
                ├─'*' ──────────────────┤                 └─system name─┘
                ├─EZECONVT──────────────┤
                ├─NONE──────────────────┤
                └─BINARY────────────────┘
```

```
►──┬─────────────────────────────────┬──┬─────────────────────────────────┬──────────────►◄
   │             ┌─GENERATION─┐      │  └─serverid=──server identifier──┘
   └─remotebind=─┴─RUNTIME────┴──────┘
```

The following attributes are ignored:
- remoteapptype
- externalname
- library
- luwcontrol

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=DCE
   location=Server1 contable=ELAAXxxx remotebind=RUNTIME serverid=Test
```

Note: Specifying the CONTABLE attribute requires the three character NLS code appropriate to your language. See “Conversion Table by Language and Platform” on page 436 for more on information on specifying the CONTABLE attribute.

DIRECT Protocol

Advantages of IPC and DIRECT Protocols

The protocols IPC and DIRECT provide a method for client programs to call server programs where both reside on the same system. A local call from the client program to the server program might be used, but there are advantages to making this call remotely via the VisualAge Generator middleware in certain cases. Futhermore, these advantages differ for the provided protocols, IPC and DIRECT, and are described below.

Calls to local C++ applications for database access from Smalltalk GUI applications where the GUI application might run in the same session with other unrelated GUI applications that require database access: GUI applications begin VisualAge Generator communications sessions when calling server programs. When the IPC protocol is used, each call from a different GUI would be made under a different process, each of which can have its own database connection and unit of work. This would not be possible if the client program used a local call to access the server program.

If two GUI applications need to be in the same unit of work, start the second GUI from the first and define the second GUI with the VAGen inheritsCommSession attribute set to true in the AbtAppBldrPart or AbtAppBldrView class. This way, local servers called by both GUI applications will share the same communications session, database connection, and unit of work.

Calls to generated C++ application from Java clients where the C++ application is to run on a web server: Multiple calls might arrive simultaneously, thus the need for multiple, independent server-controlled units of work provided by the IPC protocol.

When using the DIRECT protocol, the call is similar to a local call directly from a client program to a server program except that the VisualAge Generator middleware performs the call. Because no separate process is created, the DIRECT protocol provides improved performance over the IPC protocol. The following are scenarios for using the DIRECT protocol:

Calls to a server program running under the control of VisualAge Generator test facility from Java, VisualBasic, or PowerBuilder, where the client and server resides on the same system.

Calls to a server program running under the control of Component Broker, which maintains the database connection itself. In such an environment, the server program should not run under a separate process so that the database connection cannot be inadvertently closed when the process ends.

Error Handling

The standard error handling procedures described in "Communication Error Handling" on page 22 are supported with IPC and DIRECT server calls. EZERT8 is set to the decimal value of the client access service reason code. The reason code is the same as the number portion of error message CSOnnnna as listed in the *VisualAge Generator Messages and Problem Determination Guide* document.

Configuring an AIX Client for an AIX Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via DIRECT:

```
►►—:calllink—applname=program name—linktype=—CSOCALL—remotecomtype=—DIRECT————————►◄
```

While the specification of:

```
►►—contable=—┬—conversion table name—┬——————————————————————————————————————————►◄
             ├—'*' ——————————————————┤
             ├—EZECONVT——————————————┤
             ├—NONE——————————————————┤
             └—BINARY————————————————┘
```

is permissible, data conversion is generally not necessary when the client and server reside on the same system. All other linkage table entries are ignored on a local (DIRECT) call.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=CSOCALL remotecomtype=DIRECT
```

IPC Protocol

Advantages of IPC and DIRECT Protocols

The protocols IPC and DIRECT provide a method for client programs to call server programs where both reside on the same system. A local call from the client program to the server program might be used, but there are advantages to making this call remotely via the VisualAge Generator middleware in certain cases. Futhermore, these advantages differ for the provided protocols, IPC and DIRECT, and are described below.

Calls to local C++ applications for database access from Smalltalk GUI applications where the GUI application might run in the same session with other unrelated GUI applications that require database access: GUI applications begin VisualAge Generator communications sessions when calling server programs. When the IPC protocol is used, each call from a different GUI would be made under a different process, each of which can have its own database connection and unit of work. This would not be possible if the client program used a local call to access the server program.

If two GUI applications need to be in the same unit of work, start the second GUI from the first and define the second GUI with the VAGen inheritsCommSession attribute set to true in the AbtAppBldrPart or AbtAppBldrView class. This way, local servers called by both GUI applications will share the same communications session, database connection, and unit of work.

Calls to generated C++ application from Java clients where the C++ application is to run on a web server: Multiple calls might arrive simultaneously, thus the need for multiple, independent server-controlled units of work provided by the IPC protocol.

When using the DIRECT protocol, the call is similar to a local call directly from a client program to a server program except that the VisualAge Generator middleware performs the call. Because no separate process is created, the DIRECT protocol provides improved performance over the IPC protocol. The following are scenarios for using the DIRECT protocol:

Calls to a server program running under the control of VisualAge Generator test facility from Java, VisualBasic, or PowerBuilder, where the client and server resides on the same system.

Calls to a server program running under the control of Component Broker, which maintains the database connection itself. In such an environment, the server program should not run under a separate process so that the database connection cannot be inadvertently closed when the process ends.

Error Handling

The standard error handling procedures described in "Communication Error Handling" on page 22 are supported with IPC and DIRECT server calls. EZERT8 is set to the decimal value of the client access service reason code. The reason code is the same as the number portion of error message CSOnnnna as listed in the *VisualAge Generator Messages and Problem Determination Guide* document.

Configuring an AIX Client for an AIX server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via IPC:

```
►►──:calllink──applname=program name──linktype=─CSOCALL──remotecomtype=─IPC──────────►◄
```

While the specification of:

```
►►──contable=──┬─conversion table name─┬──────────────────────────────────────────►◄
               ├─'*'───────────────────┤
               ├─EZECONVT──────────────┤
               ├─NONE──────────────────┤
               └─BINARY────────────────┘
```

is permissable, data conversion is generally not necessary when the client and server reside on the same system. All other linkage table entries are ignored on a local (IPC) call.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=CSOCALL remotecomtype=IPC
```

TCP/IP Protocol

Creating a TCP/IP Services File Entry

The client machine must have an entry for the TCP/IP Serverid added to its TCP/IP services file. For Windows, AIX, HP-UX, and Solaris the TCP/IP SERVICES file is used.

Error Handling

The standard error handling procedures described in “Communication Error Handling” on page 22 are supported with TCPIP server calls. EZERT8 is set to the decimal value of the client access service reason code. The reason code is the same as the number portion of error message CSOnnnna as listed in the *VisualAge Generator Messages and Problem Determination Guide* document.

Configuring an AIX Client for an OS/2 server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─TCPIP──────────►

►─┬──────────────────────────────────────────┬──────────────────────────────────────►
  └─contable=─┬─conversion table name─┬──┘
              ├─'*' ──────────────────┤
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬───────────────────────────────────────────┬─────────────────────────────────────►
  └─location=───system name_tcpip_hostname───┘

►─┬────────────────────────────────────────┬────────────────────────────────────────►◄
  └─serverid=───tcpip_server identifier───┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to “Conversion Table by Language and Platform” on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 remotebind=RUNTIME serverid=port1
```

Configuring an AIX Client

Configuring an AIX client for a Windows NT server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►─:calllink─applname=program name─linktype=─REMOTE─remotecomtype=─TCPIP──────────►

►─┬──────────────────────────────────────┬────────────────────────────────────────►
  └─contable=─┬─conversion table name─┬─┘
              ├─'*' ──────────────────┤
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬───────────────────────────────────────────┬───────────────────────────────────►
  └─location=──system name_tcpip_hostname───┘

►─┬─────────────────────────────────────┬─────────────────────────────────────────►◄
  └─serverid=──tcpip_server identifier──┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to "Conversion Table by Language and Platform" on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 remotebind=RUNTIME serverid=port1
```

Configuring an AIX Client for an AIX server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►─:calllink─applname=program name─linktype=─REMOTE─remotecomtype=─TCPIP──────────►
```

```
►─┬───────────────────────────────────────┬──────────────────────────────────────────►
  └─contable=─┬─conversion table name─┬─┘
              ├─'*' ──────────────────┤
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬──────────────────────────────────────────┬───────────────────────────────────────►
  └─location=──system name_tcpip_hostname────┘

►─┬─────────────────────────────────────┬───────────────────────────────────────────►◄
  └─serverid=──tcpip_server identifier──┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to "Conversion Table by Language and Platform" on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 remotebind=RUNTIME serverid=port1
```

Configuring an AIX client for an HP-UX server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─TCPIP──────────►

►─┬───────────────────────────────────────┬──────────────────────────────────────────►
  └─contable=─┬─conversion table name─┬─┘
              ├─'*' ──────────────────┤
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬──────────────────────────────────────────┬───────────────────────────────────────►
  └─location=──system name_tcpip_hostname────┘
```

```
►─┬──────────────────────────────────┬──────────────────────────────────────►◄
  └─serverid=──tcpip_server identifier─┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to "Conversion Table by Language and Platform" on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 remotebind=RUNTIME serverid=port1
```

Configuring an AIX Client for a Solaris server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►─:calllink─applname=program name─linktype=─REMOTE─remotecomtype=─TCPIP──────────►

►─┬────────────────────────────────┬──────────────────────────────────────────────►
  └─contable=─┬─conversion table name─┬─┘
              ├─'*'───────────────────┤
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬──────────────────────────────────────────┬────────────────────────────────────►
  └─location=──system name_tcpip_hostname──┘

►─┬──────────────────────────────────┬──────────────────────────────────────────►◄
  └─serverid=──tcpip_server identifier─┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to "Conversion Table by Language and Platform" on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 remotebind=RUNTIME serverid=port1
```

Configuring an AIX Client for a VM/ESA server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►─:calllink─applname=program name─linktype=─REMOTE─remotecomtype=─TCPIP──────────►

►─┬─────────────────────────────────────┬────────────────────────────────────────►
  └─contable=─┬─conversion table name─┬─┘
              ├─'*' ──────────────────┤
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬──────────────────────────────────────────────┬───────────────────────────────►
  └─location=──system name_tcpip_hostname───────┘

►─┬─────────────────────────────────────────┬─────────────────────────────────►◄
  └─serverid=──tcpip_server identifier──────┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to “Conversion Table by Language and Platform” on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 remotebind=RUNTIME serverid=port1
```

Configuring an AIX Server

Summary Table of Valid Clients and Protocols

Table 5. Valid Clients and Protocols for VisualAge Generator Servers on the AIX Platform

Server Platform	Client Platforms				
	OS/2	Windows 95 and Windows 98	Windows NT	AIX	Solaris
AIX	TCP/IP, DCE	TCP/IP, DCE	TCP/IP, DCE	TCP/IP, DCE, IPC, DIRECT	TCP/IP, IPC, DIRECT

DCE Protocol

List of Valid Clients

- OS/2 (GUI, C++, ITF)
- Windows 95 and Windows 98 (GUI)
- Windows NT (GUI, C++, ITF)
- AIX (C++)

Identifying the Server Location

The server location is determined by the bindings advertised in the DCE CDS database. The bindings are found in the DCE CDS location object located at /.:/Servers/VAGenerator/SERVERID/LOCATION. Identify the server location to the client by specifying the location identifier in the `location` linkage table attribute in the entry for the server program, or set the location dynamically in the client program at runtime using EZELOC.

Linkage Table Attributes for Generating Server Programs

To generate native OS/2, AIX, or Windows NT server programs, no linkage table entry is required. If the server program is going to make remote calls (it is the second tier of a three tier client/server system) then a linkage table needs to be provided at generation time just as if it was a client program.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=DCE
   location=Server1 remotebind=GENERATION serverid=Test
```

DCE CDS Entries Required for VisualAge Generator DCE Servers

All VisualAge Generator DCE CDS entries are placed under the /.:/Servers/VAGenerator directory. Each serverid must have a directory, /.:/Servers/VAGenerator/serverid.

Note: It might be helpful to equate serverid with an application system and location with a server or group of servers that process requests for the application system.

Starting the DCE Server Program

The VisualAge Generator DCE server program, CSODCES, is started on the remote host machine and listens for incoming DCE requests and processes them. On AIX systems, the current userid must either be root or have access to the DCE keytab file. CSODCES takes one optional parameter and one required parameter at startup. The required parameter is the name of the configuration file to be used in starting up the DCE server. The optional parameter specifies the type of cleanup the server does when it is terminated.

```
csodces [-c | -d ] filename
```

The configuration file contains the following:

- The principal name to be used by the VisualAge Generator DCE server for DCE access and authorization
- The advertising location and serverid names of the DCE server (specified in the `location` and `serverid` attributes of the client's linkage table)
- The DCE ACL object to be used for client authorization
- The called server programs that it will be processing requests for (specified in the applname attribute of the client's linkage table).

The programs are divided into those that require authorization checking (SECURE PROGRAMS) and those that do not require authorization checking (PUBLIC PROGRAMS). Authorization checking has an impact on server performance, so only use authorization when required.

The following example shows the configuration file:

```
DCEprincipal=vgserve1
LOCATION=Server1
SERVERID=Test
DCEACLobject=/.:/Servers/VAGenerator/Test/Server1
SECURE PROGRAMS=
SECURE1
SECURE2
PUBLIC PROGRAMS=
ELACVP5
DTCALL2
MAXSIZE
PARMSRV
PRMSRV2
```

The cleanup parameter is used when there is more than one server using a serverid/location pair as its advertising location. With the -c option, which is the default, when the server terminates it will remove its entry from the RPC mapping, DCE runtime, and DCE CDS. If there are multiple servers advertising at a serverid/location location and one of the servers removes its entry, then all of the servers will lose their entries. To prevent all the servers from losing their entries, use the -d parameter which will only remove the entry from the RPC mapping when the server terminates. There should always be one server which is started without the -d parameter to ensure that all entries for the DCE servers are cleaned up after they are terminated. The server that was started without the -d parameter should be terminated last.

DIRECT Protocol

List of Valid Clients

- AIX (C++)
- Solaris (C++)

Advantages of IPC and DIRECT Protocols

The protocols IPC and DIRECT provide a method for client programs to call server programs where both reside on the same system. A local call from the client program to the server program might be used, but there are advantages to making this call remotely via the VisualAge Generator middleware in certain cases. Futhermore, these advantages differ for the provided protocols, IPC and DIRECT, and are described below.

Calls to local C++ applications for database access from Smalltalk GUI applications where the GUI application might run in the same session with other unrelated GUI applications that require database access: GUI applications begin VisualAge Generator communications sessions when calling server programs. When the IPC protocol is used, each call from a different GUI would be made under a different process, each of which can have its own database connection and unit of work. This would not be possible if the client program used a local call to access the server program.

If two GUI applications need to be in the same unit of work, start the second GUI from the first and define the second GUI with the VAGen inheritsCommSession attribute set to true in the AbtAppBldrPart or AbtAppBldrView class. This way, local servers called by both GUI applications will share the same communications session, database connection, and unit of work.

Calls to generated C++ application from Java clients where the C++ application is to run on a web server: Multiple calls might arrive simultaneously, thus the need for multiple, independent server-controlled units of work provided by the IPC protocol.

When using the DIRECT protocol, the call is similar to a local call directly from a client program to a server program except that the VisualAge Generator middleware performs the call. Because no separate process is created, the DIRECT protocol provides improved performance over the IPC protocol. The following are scenarios for using the DIRECT protocol:

Calls to a server program running under the control of VisualAge Generator test facility from Java, VisualBasic, or PowerBuilder, where the client and server resides on the same system.

Calls to a server program running under the control of Component Broker, which maintains the database connection itself. In such an environment, the server program should not run under a separate process so that the database connection cannot be inadvertently closed when the process ends.

Identifying a C++ Server Location

When using the IPC or DIRECT protocols, server programs are located using the LIBPATH environment variable. You need to ensure that the server DLL directory is specified in the LIBPATH.

IPC Protocol

List of Valid Clients

- AIX (C++)
- Solaris (C++)

Advantages of IPC and DIRECT Protocols

The protocols IPC and DIRECT provide a method for client programs to call server programs where both reside on the same system. A local call from the client program to the server program might be used, but there are advantages to making this call remotely via the VisualAge Generator middleware in certain cases. Futhermore, these advantages differ for the provided protocols, IPC and DIRECT, and are described below.

Calls to local C++ applications for database access from Smalltalk GUI applications where the GUI application might run in the same session with other unrelated GUI applications that require database access: GUI applications begin VisualAge Generator communications sessions when calling server programs. When the IPC protocol is used, each call from a different GUI would be made under a different process, each of which can have its own database connection and unit of work. This would not be possible if the client program used a local call to access the server program.

If two GUI applications need to be in the same unit of work, start the second GUI from the first and define the second GUI with the VAGen inheritsCommSession attribute set to true in the AbtAppBldrPart or AbtAppBldrView class. This way, local servers called by both GUI applications will share the same communications session, database connection, and unit of work.

Calls to generated C++ application from Java clients where the C++ application is to run on a web server: Multiple calls might arrive simultaneously, thus the need for multiple, independent server-controlled units of work provided by the IPC protocol.

When using the DIRECT protocol, the call is similar to a local call directly from a client program to a server program except that the VisualAge Generator middleware performs the call. Because no separate process is created, the DIRECT protocol provides improved performance over the IPC protocol. The following are scenarios for using the DIRECT protocol:

Calls to a server program running under the control of VisualAge Generator test facility from Java, VisualBasic, or PowerBuilder, where the client and server resides on the same system.

Calls to a server program running under the control of Component Broker, which maintains the database connection itself. In such an environment, the server program should not run under a separate process so that the database connection cannot be inadvertently closed when the process ends.

Identifying the Server Location

When using the IPC or DIRECT protocols, server programs are located using the LIBPATH environment variable. You need to ensure that the server DLL directory is specified in the LIBPATH.

TCP/IP Protocol

List of Valid Clients

- OS/2 (GUI, C++, ITF)
- Windows 95 and Windows 98 (GUI)
- Windows NT (GUI, C++, ITF)
- AIX (C++)
- Solaris (C++)

Identifying a C++ Server Location

The TCP/IP support servers locate the server programs via the LIBPATH environment variable, so you need to ensure that the server DLL directory is specified in the LIBPATH.

Server Program Set Up and Operation

The server uses a configuration file for specifying the TCP/IP service name to listen on. The configuration file is optional as there are predefined default values that will be used.

Start the TCP/IP server by issuing the command:

```
CSOTCPS "config_filename"
```

The configuration file is located via the following search order:
1. File specified on the command line
2. The file named CSO.INI in the directory specified by the CSODIR environment variable.
3. The file named CSO.INI in the current directory

The search ends when the first one of the above conditions is met. Once a file is identified, the contents of the file are used if possible. If the file cannot be opened for any reason, a warning message is printed on the console and the default values are used. CSOTCPS will display the configuration filename and values being used prior to starting to listen for incoming requests. The default values are:

```
TCP/IP service name: VAGenerator
```

Where:

TCP/IP service name	Specifies the service name that will be used to look up the TCP/IP port number that CSOTCPS will listen on for incoming requests. This entry is case sensitive and must match an entry in the TCP/IP services file. For Windows NT, AIX, and HP-UX the TCP/IP SERVICES file is used.

Sample TCP/IP Entries from CSO.INI File: The following TCP/IP entries are located in the sample CSO.INI file. All entries must start in column 1. Comments are allowed within the file by placing a semicolon (;) in column 1 of the comment line. You only need to specify an entry if you wish to override the default value. If duplicate entries are specified in the file, the last entry in the file is used (just like environment variables in the config.sys file).

```
tcp_service_name=VAGenerator
```

Chapter 4. CICS for AIX Platform

How to use this chapter

This chapter contains platform and protocol information necessary to configure a client program or a server program. The chapter is divided into configuration sections: client programs and server programs. Each configuration section is organized by protocol. Each protocol section contains information organized by target platforms.

Use the following method to locate the information you need:

1. Verify that this chapter is appropriate to your runtime environment.

   Ensure that the runtime environment of your program matches the platform on which this chapter is based. Turn to the appropriate chapter if the runtime environment of your program does not match the platform named in the title of this chapter.
2. Determine which type of program you want to configure: client program or server program.
3. Determine which configuration section in the chapter is appropriate for your program.
4. Find the intended target platform of the program you are configuring in the Summary Table of Valid Clients and Protocols or Summary Table of Valid Servers and Protocols.

   If you are configuring a client program, find the target platform of the intended server program in the column of valid platforms.

   If you are configuring a server program, find the target platform of the intended client program in the row of valid platforms.
5. Chose a protocol based on the information in the table.

   If you are configuring a client program, find the cell in the table where the row of the intended target platform intersects the column of the client platform. Select a protocol from the list of available protocols for this client/server combination.

   If you are configuring a client program, find the cell in the table where the column of the intended target platform intersects the row of the server platform. Select a protocol from the list of available protocols for this client/server combination.
6. Determine the name of the appropriate protocol section within the configuration section.

Turn to the part of the configuration section indicated by the protocol name. Here you will find information common to all target platforms valid for this protocol.

7. Determine the name of the appropriate target platform section within the protocol section.

   You will find platform specific information for the chosen protocol in the section named for your intended target platform.

Example

```
Chapter: CICS for AIX Platform
Type of program to configure: Client program
Configuration section: Configuring a CICS for AIX client
Intended target platform: CICS for MVS/ESA
(i.e. a CICS for MVS/ESA server program)
Chosen protocol: CICS DPL
Protocol section: CICS DPL Protocol
Target platform section: Configuring a CICS for AIX Client for a
CICS for MVS/ESA Server
```

You will encounter syntax diagrams in this chapter. The diagrams are used to show programming syntax as follows:

```
►►──:RequiredTerm──RequiredTerm=RequiredValue──RequiredTerm=─┬─OptionalValue─┬──────────►
                                                             └─OptionalValue─┘

►──┬──────────────────────────────────────┬──┬─────────────────────────────────┬──────────►
   │                 ┌─DefaultValue──┐    │  └─OptionalTerm=RequiredValue──────┘
   └─OptionalTerm=───┴─OptionalValue─┴────┘

►──┬──────────────────────────────────────┬────────────────────────────────────────────►◄
   └─OptionalTerm=───┬─OptionalValue─┬────┘
                     └─OptionalValue─┘
```

RequiredTerm	A required term is a term that must be specified. Required terms are depicted in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required terms.
RequiredValue	A required value is a value that must be specified for a term. A required value is shown in italics in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required values.
OptionalTerm	An optional term is a term that can be

	specified. An optional term is depicted in syntax diagrams in an alternate path from the preceding term.
OptionalValue	An optional value is a value that can be specified for a term. An optional value is shown in italics in syntax diagrams in an alternate path from the preceding term along with other optional values or a default value.
DefaultValue	A default value is the value that is specified by default for a term. A default value is shown in italics in syntax diagrams in an alternate path above other optional values.

Configuring a CICS for AIX Client

Summary Table of Valid Servers and Protocols

Table 6. Valid Servers and Protocols for VisualAge Generator Clients on the CICS for AIX Platform

Server Platforms	Protocols for CICS for AIX Platform
CICS for AIX	CICS DPL
CICS for MVS/ESA	CICS DPL
CICS for OS/2	CICS DPL
CICS for Solaris	CICS DPL
CICS for VSE/ESA	CICS DPL
CICS for Windows NT	CICS DPL

CICS DPL Protocol

User Authentication

The user associated with the CICS client transaction must be authorized to run the client transaction and the server transaction. The user ID and password are identified by the CICS transaction manager on the client system.

Setting Up Communication Links

The communication link between the client and server systems must be defined to the CICS systems to allow a distributed program link from one system to the other. Refer to the CICS intercommunication manual for your CICS systems for more information.

Controlling the Unit of Work

Use the `luwcontrol` linkage table attribute to indicate whether the server updates are automatically committed on return or the client controls the unit of work. If the client is controlling the unit of work, subsequent server calls

must go to the same system and transaction under client-controlled unit of work until a commit or roll back is requested. The server cannot issue EZECOMIT or EZEROLLB calls if client-controlled unit of work is specified.

Data Format Conversion

Code format conversion is performed on the server call as specified in the `contable` linkage attribute table and in EZECONVT. Code is converted based on the structure of the arguments that are specified on the CALL statement in the client program.

Error Handling

The standard error handling procedures described below are supported with CICS DPL. If the DPL is not successful for any reason, including unavailability of the communication link, error information is returned with CICS in the CICS EIB (EXEC interface block).

If the REPLY option is not specified on the CALL statement, the calling program ends with error messages. If REPLY is specified, no messages are written or logged, and EZERT8 is set to one of the following values:

Value	Meaning
00000000	Successful call and return
00000204	Program name not valid
00000207	System identifier not valid
00000208	Link out of service
ffrrrrrr	Other CICS error where ff is the hexadecimal representation of EIBFN byte 0, and rrrrrr is the hexadecimal representation of EIBRCODE bytes 0–2.

Configuring a CICS for AIX Client for a CICS for OS/2 Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►──:calllink──applname=program name──linktype=─REMOTE──┬──────────────────────┬──────────►
                                                        └─parmform=─COMMDATA───┘

►──┬───────────────────────────────────────┬──────────────────────────────────────────────►
   │              ┌─GENERATION─┐           │
   └─remotebind=──┴────────────┴─RUNTIME───┘
```

```
►─┬───────────────────────────────────────┬─┬───────────────────────────┬─────────────────►
  └─contable=─┬─conversion table name─┬─┘ └─location=─┬─EZELOC──────┬─┘
              ├─'*' ──────────────────┤               └─system name─┘
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬─────────────────────────┬─┬──────────────────────────────┬────────────────────────────►
  │             ┌─CLIENT─┐  │ └─remoteapptype=─┬─VG────┬─────┘
  └─luwcontrol=─┴─SERVER─┴──┘                  ├─NONVG─┤
                                               └─ITF───┘

►─┬─────────────────────────────────┬───────────────────────────────────────────────────►◄
  └─serverid=──server identifier────┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CSO7 contable=BINARY LUWCONTROL=CLIENT
```

Configuring a CICS for AIX Client for a CICS for Windows NT Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►─:calllink──applname=program name──linktype=─REMOTE─┬────────────────────┬────────────►
                                                      └─parmform=─COMMDATA─┘

►─┬──────────────────────────────────────┬────────────────────────────────────────────────►
  │             ┌─GENERATION─┐           │
  └─remotebind=─┴────────────┴─RUNTIME───┘

►─┬───────────────────────────────────────┬─┬───────────────────────────┬─────────────────►
  └─contable=─┬─conversion table name─┬─┘ └─location=─┬─EZELOC──────┬─┘
              ├─'*' ──────────────────┤               └─system name─┘
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘
```

```
►──┬─────────────────────────────┬──┬───────────────────────────┬──────────────────►
   │              ┌─CLIENT─┐     │  └─remoteapptype=─┬─VG────┬─┘
   └─luwcontrol=──┴─SERVER─┴─────┘                   ├─NONVG─┤
                                                     └─ITF───┘

►──┬─────────────────────────────────┬───────────────────────────────────────────►◄
   └─serverid=──server identifier────┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CS07 contable=BINARY LUWCONTROL=CLIENT
```

Configuring a CICS for AIX Client for a CICS for AIX Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►──:calllink──applname=program name──linktype=─REMOTE─┬─────────────────────┬────►
                                                       └─parmform=─COMMDATA──┘

►──┬────────────────────────────────────────┬────────────────────────────────────►
   │              ┌─GENERATION─┐            │
   └─remotebind=──┴────────────┴──RUNTIME───┘

►──┬──────────────────────────────────────┬──┬─────────────────────────────┬─────►
   └─contable=─┬─conversion table name─┬──┘  └─location=─┬─EZELOC──────┬──┘
               ├─'*'───────────────────┤                 └─system name─┘
               ├─EZECONVT──────────────┤
               ├─NONE──────────────────┤
               └─BINARY────────────────┘

►──┬─────────────────────────────┬──┬───────────────────────────┬──────────────────►
   │              ┌─CLIENT─┐     │  └─remoteapptype=─┬─VG────┬─┘
   └─luwcontrol=──┴─SERVER─┴─────┘                   ├─NONVG─┤
                                                     └─ITF───┘
```

```
►─┬────────────────────────────────┬──────────────────────────────────────────────────►◄
  └─serverid=──server identifier───┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CS07 contable=NONE LUWCONTROL=CLIENT
```

Configuring a CICS for AIX Client for a CICS for MVS/ESA Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►─:calllink─applname=program name─linktype=─REMOTE─┬──────────────────────┬─────────►
                                                    └─parmform=─COMMDATA───┘

►─┬───────────────────────────────────────────┬──────────────────────────────────────►
  │              ┌─GENERATION─┐               │
  └─remotebind=──┴────────────┴─RUNTIME───────┘

►─┬──────────────────────────────────────────┬─┬──────────────────────────────┬───────►
  └─contable=─┬─conversion table name─┬──────┘ └─location=─┬─EZELOC──────┬─────┘
              ├─'*'───────────────────┤                   └─system name─┘
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬──────────────────────────────┬─┬─────────────────────────────────┬────────────────►
  │              ┌─CLIENT─┐      │ └─remoteapptype=─┬─VG─────┬───────┘
  └─luwcontrol=──┴─SERVER─┴──────┘                  ├─NONVG──┤
                                                    └─ITF────┘

►─┬────────────────────────────────┬──────────────────────────────────────────────────►◄
  └─serverid=──server identifier───┘
```

The following attributes are ignored:
- externalname
- library

- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CS07 contable=ELACNENU LUWCONTROL=CLIENT
```

Configuring a CICS for AIX Client for a CICS for VSE/ESA Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►─:calllink─applname=program name─linktype=─REMOTE──┬─────────────────────┬──────►
                                                      └─parmform=─COMMDATA──┘

►─┬─────────────────────────────────────┬──────────────────────────────────────────►
  │                ┌─GENERATION─┐       │
  └─remotebind=────┴────────────┴─RUNTIME┘

►─┬─────────────────────────────────────┬─┬─────────────────────────────┬──────────►
  └─contable=─┬─conversion table name─┬─┘ └─location=─┬─EZELOC──────┬──┘
              ├─'*' ──────────────────┤              └─system name─┘
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬───────────────────────────┬─┬──────────────────────────────┬───────────────────►
  │               ┌─CLIENT─┐  │ └─remoteapptype=─┬─VG──────┬───┘
  └─luwcontrol=───┴─SERVER─┴──┘                  ├─NONVG───┤
                                                 └─ITF─────┘

►─┬──────────────────────────────────┬────────────────────────────────────────────►◄
  └─serverid=──server identifier─────┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CS07 contable=ELACNENU LUWCONTROL=CLIENT
```

Configuring a CICS for AIX Client for a CICS for Solaris Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►──:calllink──applname=program name──linktype=─REMOTE───────────────────────────►
                                                      └parmform=─COMMDATA──┘

►─┬───────────────────────────────────┬──────────────────────────────────────────►
  │               ┌GENERATION─┐       │
  └remotebind=────┴───────────┴RUNTIME┘

►─┬──────────────────────────────────┬─┬──────────────────────────────┬──────────►
  └contable=──┬conversion table name─┤ └location=──┬EZELOC─────────┬─┘
              ├'*'───────────────────┤             └system name────┘
              ├EZECONVT──────────────┤
              ├NONE──────────────────┤
              └BINARY────────────────┘

►─┬─────────────────────────┬─┬────────────────────────────────┬─────────────────►
  │             ┌CLIENT─┐   │ └remoteapptype=──┬VG────┬────────┘
  └luwcontrol=──┴SERVER─┴───┘                  ├NONVG─┤
                                               └ITF───┘

►─┬──────────────────────────────────┬──────────────────────────────────────────►◄
  └serverid=───server identifier─────┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CS07 contable=NONE LUWCONTROL=CLIENT
```

Configuring a CICS for AIX Server

Summary Table of Valid Clients and Protocols

Table 7. Valid Clients and Protocols for VisualAge Generator Servers on the CICS for AIX Platform

Server Platform	**Client Platforms**					
	OS/2 (GUI, C++, ITF)	**Windows 95 and Windows 98 (GUI)**	**Windows NT (GUI, C++, ITF)**	**AIX (C++)**	**CICS**	**Solaris (C++)**
CICS for AIX	CICS Client	CICS Client	CICS Client	CICS Client	CICS DPL	CICS Client
Notes: CICS as a Client Platform refers to: – CICS for AIX – CICS for MVS/ESA – CICS for OS/2 – CICS for Windows NT – CICS for VSE/ESA – CICS for Solaris						

CICS Client Protocol

List of Valid Clients

- OS/2 (GUI, C++, ITF)
- Windows 95 and Windows 98 (GUI)
- Windows NT (GUI, C++, ITF)
- AIX (C++)
- Solaris (C++)

Linkage Table Attributes for Generating CICS Server Programs

Specify the following calllink attributes when you generate CICS server programs:

```
►►──:calllink──applname=program name──linktype=─REMOTE──┬────────────────────┬──────►◄
                                                         └─parmform=─COMMDATA─┘
```

The following attributes are ignored:

- contable
- externalname
- library
- location
- luwcontrol
- remoteapptype
- remotebind
- remotecomtype

- serverid

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA
```

Considerations for Defining the Server Program
If the remote program performs printing, it must move the print destination into EZEDESTP.

Server Program Set Up
The server program is generated and prepared as is any other CICS program on the server system. The server transaction must be defined to CICS on the server system. The default transaction name associated with a server program is the CICS-supplied mirror transaction, CPMI.

Specifying Parameter Format
COMMDATA is the only valid parameter format (the parmform attribute on the calllink tag) that you can specify.

CICS DPL Protocol

List of Valid Clients
- CICS for AIX
- CICS for MVS/ESA
- CICS for OS/2
- CICS for Windows NT
- CICS for VSE/ESA
- CICS for Solaris

Identifying the Server Location
You can specify the server location (CICS system identifier) on the `location` linkage table attribute, or dynamically set the location in the client program at runtime using EZELOC. If the location is not specified, the default location is the system identifier associated with the server program in the CICS program definition on the client system.

Linkage Table Attributes for Generating CICS Server Programs
Specify the following calllink attributes when you generate CICS server programs:

```
►►──:calllink──applname=program name──linktype=─REMOTE──┬──────────────────────┬────────►◄
                                                        └─parmform=─COMMDATA───┘
```

The following attributes are ignored:
- contable
- externalname

- library
- location
- luwcontrol
- remoteapptype
- remotebind
- remotecomtype
- serverid

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA
```

Considerations for Defining the Server Program

If the remote program performs printing, it must move the print destination into EZEDESTP.

Server Program Set Up

The server program is generated and prepared as is any other CICS program on the server system. In addition, there must be a CICS program definition for the server program on the CICS client system containing the following information:

- Resident option = remote
- Remote system identifier
- Remote program name (optional)
- Remote transaction identifier (optional)

The server transaction must be defined to CICS on the server system. The default transaction name associated with a server program is the CICS-supplied mirror transaction, CPMI.

Chapter 5. OS/2 Platform

How to use this chapter

This chapter contains platform and protocol information necessary to configure a client program or a server program. The chapter is divided into configuration sections: client programs and server programs. Each configuration section is organized by protocol. Each protocol section contains information organized by target platforms.

Use the following method to locate the information you need:

1. Verify that this chapter is appropriate to your runtime environment.

   Ensure that the runtime environment of your program matches the platform on which this chapter is based. Turn to the appropriate chapter if the runtime environment of your program does not match the platform named in the title of this chapter.
2. Determine which type of program you want to configure: client program or server program.
3. Determine which configuration section in the chapter is appropriate for your program.
4. Find the intended target platform of the program you are configuring in the Summary Table of Valid Clients and Protocols or Summary Table of Valid Servers and Protocols.

   If you are configuring a client program, find the target platform of the intended server program in the column of valid platforms.

   If you are configuring a server program, find the target platform of the intended client program in the row of valid platforms.
5. Chose a protocol based on the information in the table.

   If you are configuring a client program, find the cell in the table where the row of the intended target platform intersects the column of the client platform. Select a protocol from the list of available protocols for this client/server combination.

   If you are configuring a client program, find the cell in the table where the column of the intended target platform intersects the row of the server platform. Select a protocol from the list of available protocols for this client/server combination.
6. Determine the name of the appropriate protocol section within the configuration section.

Turn to the part of the configuration section indicated by the protocol name. Here you will find information common to all target platforms valid for this protocol.

7. Determine the name of the appropriate target platform section within the protocol section.

   You will find platform specific information for the chosen protocol in the section named for your intended target platform.

Example

```
Chapter: OS/2 Platform
Type of program to configure: Client program
Configuration section: Configuring an OS/2 client
Intended target platform: CICS for MVS/ESA
(i.e. a CICS for MVS/ESA server program)
Chosen protocol: CICS Client
Protocol section: CICS Client Protocol
Target platform section: Configuring an OS/2 Client for a
CICS for MVS/ESA Server
```

You will encounter syntax diagrams in this chapter. The diagrams are used to show programming syntax as follows:

```
►►──:RequiredTerm──RequiredTerm=RequiredValue──RequiredTerm=─┬─OptionalValue─┬──────────►
                                                             └─OptionalValue─┘

►──┬────────────────────────────────────┬──┬──────────────────────────────┬─────────────►
   │                  ┌─DefaultValue──┐ │  └─OptionalTerm=RequiredValue─┘
   └─OptionalTerm=────┴─OptionalValue─┴─┘

►──┬────────────────────────────────────┬───────────────────────────────────────────────►◄
   └─OptionalTerm=──┬─OptionalValue─┬───┘
                    └─OptionalValue─┘
```

RequiredTerm	A required term is a term that must be specified. Required terms are depicted in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required terms.
RequiredValue	A required value is a value that must be specified for a term. A required value is shown in italics in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required values.
OptionalTerm	An optional term is a term that can be

	specified. An optional term is depicted in syntax diagrams in an alternate path from the preceding term.
OptionalValue	An optional value is a value that can be specified for a term. An optional value is shown in italics in syntax diagrams in an alternate path from the preceding term along with other optional values or a default value.
DefaultValue	A default value is the value that is specified by default for a term. A default value is shown in italics in syntax diagrams in an alternate path above other optional values.

Configuring an OS/2 Client

Summary Table of Valid Servers and Protocols

Table 8. Valid Servers and Protocols for VisualAge Generator Clients on the OS/2 Platform

Server Platforms	Protocols for OS/2 Platform
AIX	TCP/IP, DCE
CICS for AIX	CICS Client
HP-UX	TCP/IP
IMS	APPC/IMS
CICS for MVS/ESA	CICS Client, LU2
OS/2	TCP/IP, DCE, IPC, DIRECT
CICS for OS/2	CICS Client
OS/400	CA/400
Solaris	TCP/IP
CICS for Solaris	CICS Client
VM/ESA	TCP/IP
CICS for VSE/ESA	CICS Client
Windows NT	TCP/IP, DCE
Windows NT (Java)	N/A
CICS for Windows NT	CICS Client

Configuring an OS/2 Client

APPC/IMS Protocol

User Authentication

The user authentication exit provides the user ID and the password specified when the APPC session is allocated. User authentication is described in “User Authentication” on page 21. The program user must be authorized to run the transaction associated with the server call (the transaction program name used on the LU 6.2 connection). For further information, see "Identifying the Server Location".

Setting Up Communication Links

The LU 6.2 communication link between the client and server systems must be defined to the host server system and to the client products so that an LU 6.2 session can be allocated between the client and the APPC component of IMS on the host system. Refer to the IMS and communications product documentation for information on setting up the LU 6.2 connection between the client and host systems.

Controlling the Unit of Work

Each server call is an independent unit of work. Any updates made by the server are committed when the server returns to the client. Any EZECOMIT calls issued by the server are ignored. Calling EZEROLLB or a terminating error caught by Server for MVS, VSE, and VM both result in a DL/I ROLB call being issued.

Data Format Conversion

Code format conversion is performed on the server call as specified on the `contable` linkage table attribute and EZECONVT special function word. Code is converted based on the structure of the arguments specified on the CALL statement in the client program.

Error Handling

The standard error handling procedures described in “Communication Error Handling” on page 22 are supported with IMS server calls. EZERT8 is set to the decimal value of the client access service reason code. The reason code is the same as the number portion of error message CSOnnnna as listed in the *VisualAge Generator Messages and Problem Determination Guide* document.

Configuring an OS/2 Client for an IMS Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test client programs that call the server program via APPC:

```
►►:calllink—applname=program name——linktype=——REMOTE————remotecomtype=——APPCIMS————location=—┬─EZELOC──────┬──►
                                                                                            └─system name─┘
```

```
►─┬──────────────────┬──┬────────────────────┬──┬──────────────────────────────────┬──►
  └─parmform=──OSLINK─┘  └─luwcontrol=──SERVER─┘  └─contable=─┬─conversion table name─┬─┘
                                                              ├─'*'───────────────────┤
                                                              ├─EZECONVT──────────────┤
                                                              ├─NONE──────────────────┤
                                                              └─BINARY────────────────┘

►─┬─────────────────────────────┬──────────────────────────────────────────────────────►◄
  │              ┌─GENERATION─┐ │
  └─remotebind=──┴─RUNTIME────┴─┘
```

The following attributes are ignored:
- externalname
- library
- remoteapptype
- serverid

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=APPCIMS
   location=IMSSIDE parmform=oslink luwcontrol=server contable=ELACNENU
```

Client Access/400 Protocol

This chapter contains information specific to implementing client/server calls using Client Access/400 (CA/400). Be sure to read the general information in "Chapter 2. Introduction to Client/Server Processing with Synchronous Calls" on page 9 before reading this chapter.

User Authentication

If the server program accesses files or relational databases, the user identified on the client must be authorized to run the server program and to access any files or relational tables using dynamic SQL statements.

The connection will be made automatically on the first server call if it was not started prior to the call. The connection function prompts for the user ID and password. The OS/2 Communications Manager must be running before the connection can be made.

Controlling the Unit of Work

Use the `luwcontrol` linkage table attribute to indicate whether server updates are automatically committed on return or whether the client controls the unit of work. All calls to the same OS/400 from the same client session run under the same OS/400 job. If you mix server-controlled unit of work calls with client-controlled unit of work calls, be aware that any commit or rollback, client or server, issued under that job will commit or rollback all outstanding updates in effect for that job.

Data Format Conversion

Code format conversion is performed on the call to the server program as directed by the developer using the `contable` linkage table attribute and

EZECONVT. Code is converted based on the structure of the parameters specified on the server call in the client program.

Error Handling

The standard error handling procedures described in “Communication Error Handling” on page 22 are supported with Client Access/400. EZERT8 is set to the decimal value of the client access service reason code. The reason code is the same as the number portion of error message CSOnnnna as listed in the *VisualAge Generator Messages and Problem Determination Guide* document.

Any error messages logged on the server are spooled to the OS/400 user ID of the user signed on to Client Access/400 on the client system. The job log will contain all messages logged since the job was started. The client access job is a prestarted job used repeatedly by many users; therefore, the job log may contain messages from other remote commands. Move to the bottom of the job log to see the last set of error messages.

Configuring an OS/2 Client for an OS/400 Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call OS/400 server programs via CA/400:

```
►►─:calllink─applname=program name─contable=─┬─conversion table name─┬──────────►
                                             ├─'*'───────────────────┤
                                             ├─EZECONVT──────────────┤
                                             ├─NONE──────────────────┤
                                             └─BINARY────────────────┘

►─linktype=─REMOTE─library=library name─remotecomtype=─CA400──────────────────────►

►─┬────────────────────┬─┬──────────────────────────────┬─────────────────────────►
  └─parmform=─OSLINK───┘ │               ┌─applname─┐   │
                         └─externalname=─┴──────────┴───┘

►─┬──────────────────────────┬─┬───────────────────────────┬──────────────────────►
  └─location=─┬─EZELOC──────┬┘ │             ┌─CLIENT─┐    │
              └─system name─┘  └─luwcontrol=─┴─SERVER─┴────┘

►─┬─────────────────────────┬─┬───────────────────────────────┬──────────────────►◄
  └─remoteapptype=─┬─VG────┬┘ │             ┌─GENERATION─┐    │
                   ├─NONVG─┤  └─remotebind=─┴─RUNTIME────┴────┘
                   └─ITF───┘
```

The following attributes are ignored:
- serverid

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CA400
contable=ELACNENU library=ELACVP5 parmform=OSLINK location=SILVER6
```

Client Access/400 Set Up: Client programs use the Client Access/400 Remote Command/Distributed Program Call APIs to call the generated server program. Ensure that you have done the following before attempting to run the VisualAge Generator client program:

- Install the appropriate Client Access/400 product on the OS/2client system
- Start the communication manager. Client Access/400 will then attempt to start the communication link when the VisualAge Generator program issues the remote command.

Your Client Access/400 set up determines whether LU 6.2 or TCP/IP is used for communication.

For further information, refer to the following manual:

- *Client Access/400 Optimized for OS/2 - Getting Started*, SC41-3510.

Reducing the Amount of Memory Required by Client Access/400 Optimized for OS/2: This section describes how to reduce the amount of memory required by Client Access/400 Optimized for OS/2 and allows the Distributed Program Call APIs to call the generated server. When the following recommendations are applied, VisualAge Generator clients and Client Access/400 run well on a 16MB, 33 MHz 486 system with OS/2.

1. Modify your config.sys file by completing the following steps:
   a. Remove or REM out the following lines: (note the // are comments describing the function of these lines they do not appear in config.sys)

      ```
      - CALL=F:\CAOS2\PRPP2.EXE F:\CAOS2   // Update locked file processor
      - RUN=F:\CAOS2\DMISL.EXE             // DMI Service Layer
      - IFS=F:\CAOS2\CWBBS.IFS             // Network drive file system
      - DEVICE=F:\CAOS2\CWBNPRDR.SYS       // Network printer redirector
      - IFS=F:\CAOS2\CWBNPFS.IFS           // Network printer file system
      - RUN=F:\CAOS2\CWBDAEMN.EXE          // Network daemon
      - IFS=F:\CAOS2\EHNSFL0.DLL           // V2 network drive file system
      - DEVICE=F:\CAOS2\EHNPCPDD.SYS       // DOS/Windows comm support
      - DEVICE=F:\CAOS2\EHNPCVDD.SYS       // DOS/Windows comm support
      ```

      It is highly recommended that you add the following statement:

      ```
      SET RESTARTOBJECTS=STARTUPFOLDERSONLY
      ```

   b. Reboot after modifying your config.sys.
2. Modify CASERV.CMD file located in the CAOS2 directory by completing the following steps:

   a. Remove or REM out the following lines: (note the // are comments describing the function of these lines they do not appear in CASERV.CMD)

```
      - CWBLOG.EXE START    // Service history logging
      - STARTRTR.EXE /G     // 16-bit router
      - VDMSERV.EXE /Z      // DOS/Windows box communication server
      - CWBBSTRT.EXE        // Network drives daemon
      - DETACH CWBMGD.EXE   // Client Management daemon
```

3. Stop CM Attach Manager by completing the following steps:
   a. From the Client Access Folder, select the Client Access/400 Components folder; then select Communications Manager; then select Subsystem Management.
   b. From APPC attach Manager, select Service; then select Stop Normal
   c. Deactivate links to other systems you do not need
4. The following changes are also possible:
   - Remove any unnecessary device drivers from CONFIG.SYS
   - Use LAPS or MPTS configuration to remove unneeded protocol stacks. DPC only requires 802.2 for SNA connections.
   - Remove DOS/Windows box device drivers from CONFIG.SYS, if DOS or Windows support is not required. You can also remove this through the OS/2 selective install feature. Also, you can set PROTECTONLY=YES, which will return about 1M to OS/2 instead of reserving it for DOS programs.

```
     rem // DOS stuff not needed if PROTECTONLY=YES...
     PROTECTONLY=YES
     rem SHELL=D:\OS2\MDOS\COMMAND.COM D:\OS2\MDOS /P
     rem FCBS=16,8
     RMSIZE=0
     rem DEVICE=D:\OS2\MDOS\VW32S.SYS
     rem DEVICE=D:\OS2\MDOS\VWIN.SYS
     rem DEVICE=D:\OS2\MDOS\VW32S.SYS
     rem DEVICE=D:\OS2\MDOS\VVGA.SYS
     rem DEVICE=D:\OS2\MDOS\VXGA.SYS
```

   - Preallocate the SWAP file.

     The initial size of the OS/2 swap file can be set in the CONFIG.SYS file. It has a default of 2048. Depending on the amount of memory on your system and the number of programs you normally run, the final size of the swap file will be considerably larger. Setting the initial size so that it is relatively close to the final setting prevents resizing the swap file.

     Find the swap file entry in your CONFIG.SYS file SWAPPATH=d:\OS2\SYSTEM 2048 2048 and change it to your final setting.

     For example:

```
SWAPPATH=d:\OS2\SYSTEM 2048 16048
```

- Minimize some OS/2 settings: (note the // are comments describing the function of these lines they do not appear in config.sys)

```
      IFS=D:\OS2\HPFS.IFS /CACHE:64   //minimize disk cache to 64K
      BUFFERS=30                      //minimize file buffers
      rem DISKCACHE=D,LW              //DISKCACHE is a FAT only setting
      rem BASEDEV=IBM1FLPY.ADD        //not needed on PS/2 micro channel PC
```

- Remove or REM Network protocols and device drivers that are not needed.

  For example:

```
      rem // NetBios
      rem DEVICE=D:\IBMCOM\PROTOCOL\NETBEUI.OS2
      rem DEVICE=D:\IBMCOM\PROTOCOL\NETBIOS.OS2
      rem // Lan Requester
      rem DEVICE=D:\IBMLAN\NETPROG\RDRHELP.200
      rem IFS=D:\IBMLAN\NETPROG\NETWKSTA.200 /I:D:\IBMLAN /N
      rem RUN=D:\IBMLAN\NETPROG\LSDAEMON.EXE
      rem // CID install
      rem DEVICE=E:\ODDCS\SRVIFS.SYS
      rem IFS=E:\ODDCS\SRVIFSC.IFS *
      rem // TCP/IP
      rem RUN=E:\TCPIP\BIN\CNTRL.EXE
      rem IFS=E:\TCPIP\BIN\NFS200.IFS
      rem DEVICE=E:\TCPIP\BIN\VDOSTCP.VDD
      rem DEVICE=E:\TCPIP\BIN\VDOSTCP.SYS
      rem RUN=E:\TCPIP\BIN\VDOSCTL.EXE
      rem DEVICE=E:\TCPIP\BIN\SNACKETS.SYS
      rem DEVICE=E:\TCPIP\BIN\SXIFNDIS.SYS
```

- Reboot after changes are applied to your config.sys file.

CICS Client Protocol

User Authentication

The user authentication exit (see "User Authentication" on page 21) provides the user ID and the password specified on the ECI call. The program user must be authorized to run the transaction associated with the server call.

The user exit can return NULLs for the userid and password. The default exit returns the contents of the environment variables CSOUID and CSOPWD as the userid and password. If nulls are specified on the ECI call, the CICS Client determines the user ID and password in a system-dependent fashion. Refer to the CICS Client documentation for your environment for further information.

Setting Up Communication Links

The communication link between the client and server systems must be defined to the CICS server system and client products to allow an ECI call to flow from a CICS Client product to server systems. CICS Client, the communications software, is installed and configured on the client. Refer to CICS CLIENT documentation. The CICS server environment must have a "listener" defined and requires other entries if remote programs are called. Refer to CICS documentation for more information. Refer to the CICS intercommunication documentation for your CICS systems and client products for additional information.

Identifying the CICS Transaction for the Server

The CICS transaction name associated with the server program is specified in the `serverid` linkage table attribute. If not specified, the default transaction is the CICS-supplied mirror transaction, CPMI.

Controlling the Unit of Work

Extended Units of Work: Multiple synchronous calls to CICS servers can be issued from the same client. You can use the extended unit of work feature of ECI to include several calls to the same system within the same unit of work by specifying `luwcontrol=CLIENT` in the linkage table for the server programs. For CICS servers, the default value for LUWCONTROL is client unit of work. The extended unit of work ends when the client program calls EZECOMIT or EZEROLLB, which results in an ECI call to commit or roll back any extended transactions that are currently active.

A separate ECI extended unit of work (CICS transaction) is started for each unique `serverid/location` pair. Servers on the same system running under the same SERVERID (transaction name) are part of the same CICS extended transaction. A client EZECOMIT or ROLLBACK call ends all the extended transactions currently in effect for the client.

The server cannot issue EZECOMIT or EZEROLLB calls if the client unit of work was in effect for the server call.

Server Unit of Work: You can specify `luwcontrol=SERVER` in the linkage table. With this specification, each server call is a separate unit of work. The server program can issue commit or rollback requests as well.

Data Format Conversion

Code format conversion is performed on the server call as specified on the `contable` linkage table attribute and EZECONVT special function word. Code is converted based on the structure of the arguments specified on the CALL statement in the client program.

Error Handling

The standard error handling procedures described in "Communication Error Handling" on page 22 are supported with CICS ECI. EZERT8 is set to the decimal value of the client access service reason code. The reason code is the same as the number portion of error message CSOnnnna as listed in the *VisualAge Generator Messages and Problem Determination Guide* document.

Specifying Parameter Format

Parameter format (the `parmform` attribute on the calllink tag) can be specified as COMMDATA (data values passed in the COMMAREA) or COMMPTR (pointers passed in the COMMAREA).

Using COMMPTR Format: Use COMMPTR for calling COBOL subroutines that were developed to use the non-VAGen parameter passing convention that uses pointers in the CICS COMMAREA. When the pointer format is used, the generated client program calls an intermediate catcher program (ELACSV, provided with VisualAge Generator host services) via the ECI instead of calling the server program directly. The catcher program receives the parameter data from the client, builds a pointer list in the COMMAREA that points to the individual parameters, and links to the server program. A program processing table (PPT) must be defined for the ELACSV program on the CICS server system.

COMMPTR format is only supported for MVS CICS, CICS for VSE/ESA, and CICS OS/2 servers.

Using COMMDATA Format: Use COMMDATA for calling generated programs and for calling COBOL subroutines where possible. COMMDATA provides better performance in calling remote programs. With COMMDATA, the server program is called directly without going through the catcher program.

Configuring an OS/2 Client for a CICS for OS/2 Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─CICSCLIENT──────►

►──┬─────────────────────────────────────────┬──┬─────────────────────────────┬─────────►
   └─contable=─┬─conversion table name─┬─────┘  └─location=─┬─EZELOC──────┬──┘
               ├─'*' ──────────────────┤                    └─system name─┘
               ├─EZECONVT──────────────┤
               ├─NONE──────────────────┤
               └─BINARY────────────────┘
```

```
►─┬─────────────────────────────┬──┬───────────────────────────┬──────────►
  │               ┌─CLIENT─┐    │  └─parmform=─┬─OSLINK─────┬─┘
  └─luwcontrol=───┴─SERVER─┴────┘              ├─COMMPTR────┤
                                               ├─COMMDATA───┤
                                               └─CICSOSLINK─┘

►─┬────────────────────────────┬─┬─────────────────────────────────────┬──►
  └─remoteapptype=─┬─VG────┬───┘ │               ┌─GENERATION─┐      │
                   ├─NONVG─┤     └─remotebind=───┴─RUNTIME────┴──────┘
                   └─ITF───┘

►─┬──────────────────────────────────┬───────────────────────────────────►◄
  └─serverid=──server identifier─────┘
```

The following attributes are ignored:
- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST
```

Configuring an OS/2 Client for a CICS for Windows NT Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►─:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─CICSCLIENT──────►

►─┬──────────────────────────────────────┬─┬───────────────────────────────┬──►
  └─contable=─┬─conversion table name─┬──┘ └─location=─┬─EZELOC──────┬────┘
              ├─'*'───────────────────┤               └─system name─┘
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬─────────────────────────────┬──┬───────────────────────────┬──────────►
  │               ┌─CLIENT─┐    │  └─parmform=─┬─OSLINK─────┬─┘
  └─luwcontrol=───┴─SERVER─┴────┘              ├─COMMDATA───┤
                                               └─CICSOSLINK─┘
```

```
►──┬─────────────────────────────┬──┬───────────────────────────────────┬──────────────►
   └─remoteapptype=──┬─VG────────┘  └─remotebind=──┬─GENERATION──┬─────┘
                     ├─NONVG─┤                     └─RUNTIME─────┘
                     └─ITF───┘

►──┬───────────────────────────────────┬─────────────────────────────────────────────►◄
   └─serverid=──server identifier──────┘
```

The following attributes are ignored:
- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST
```

Configuring an OS/2 Client for a CICS for AIX Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─CICSCLIENT──────►

►──┬──────────────────────────────────────┬──┬──────────────────────────────┬──────────►
   └─contable=──┬─conversion table name──┬┘  └─location=──┬─EZELOC──────┬──┘
                ├─'*' ───────────────────┤                └─system name─┘
                ├─EZECONVT───────────────┤
                ├─NONE───────────────────┤
                └─BINARY─────────────────┘

►──┬────────────────────────────────┬──┬──────────────────────────────┬────────────────►
   │                ┌─CLIENT──┐     │  └─parmform=──┬─OSLINK──────┬───┘
   └─luwcontrol=────┴─SERVER──┴─────┘               ├─COMMDATA────┤
                                                    └─CICSOSLINK──┘

►──┬─────────────────────────────┬──┬───────────────────────────────────┬──────────────►
   └─remoteapptype=──┬─VG────────┘  └─remotebind=──┬─GENERATION──┬─────┘
                     ├─NONVG─┤                     └─RUNTIME─────┘
                     └─ITF───┘

►──┬───────────────────────────────────┬─────────────────────────────────────────────►◄
   └─serverid=──server identifier──────┘
```

The following attributes are ignored:
- externalname

- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST contable=BINARY
```

Configuring an OS/2 Client for a CICS for MVS/ESA Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►─:calllink─applname=program name─linktype=─REMOTE─remotecomtype=─CICSCLIENT─────►

►─┬──────────────────────────────────────────┬─┬───────────────────────────────┬─►
  └─contable=─┬─conversion table name─┬──────┘ └─location=─┬─EZELOC──────┬──────┘
              ├─'*'───────────────────┤                   └─system name─┘
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬─────────────────────────────┬─┬─────────────────────────────┬────────────────►
  │              ┌─CLIENT─┐     │ └─parmform=─┬─OSLINK─────┬────┘
  └─luwcontrol=──┴─SERVER─┴─────┘             ├─COMMPTR────┤
                                              ├─COMMDATA───┤
                                              └─CICSOSLINK─┘

►─┬───────────────────────────────┬─┬─────────────────────────────────┬──────────►
  └─remoteapptype=─┬─VG─────┬─────┘ │               ┌─GENERATION─┐    │
                   ├─NONVG──┤       └─remotebind=───┴─RUNTIME────┴────┘
                   └─ITF────┘

►─┬──────────────────────────────────┬───────────────────────────────────────────►◄
  └─serverid=──server identifier─────┘
```

The following attributes are ignored:
- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST contable=ELACNENU
```

Configuring an OS/2 Client for a CICS for VSE/ESA Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─CICSCLIENT──────►

►──┬───────────────────────────────────────────┬──┬─────────────────────────────┬──────►
   └─contable=──┬─conversion table name─┬──────┘  └─location=──┬─EZELOC──────┬──┘
                ├─'*' ──────────────────┤                     └─system name─┘
                ├─EZECONVT──────────────┤
                ├─NONE──────────────────┤
                └─BINARY────────────────┘

►──┬─────────────────────────────┬──┬──────────────────────────────────┬──────────────►
   │                ┌─CLIENT─┐   │  └─parmform=──┬─OSLINK─────┬───────┘
   └─luwcontrol=────┴─SERVER─┴───┘               ├─COMMPTR────┤
                                                 ├─COMMDATA───┤
                                                 └─CICSOSLINK─┘

►──┬──────────────────────────────────┬──┬───────────────────────────────────┬────────►
   └─remoteapptype=──┬─VG────┬────────┘  │                ┌─GENERATION─┐     │
                     ├─NONVG─┤           └─remotebind=────┴─RUNTIME────┴─────┘
                     └─ITF───┘

►──┬──────────────────────────────────┬───────────────────────────────────────────────►◄
   └─serverid=───server identifier────┘
```

The following attributes are ignored:
- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST contable=ELACNENU
```

Configuring an OS/2 Client for a CICS for Solaris Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─CICSCLIENT──────►
```

```
►──┬───────────────────────────────────────────┬──┬──────────────────────────────┬──────────────►
   └─contable=──┬─conversion table name─┬──────┘  └─location=──┬─EZELOC──────┬──┘
                ├─'*' ──────────────────┤                      └─system name─┘
                ├─EZECONVT──────────────┤
                ├─NONE──────────────────┤
                └─BINARY────────────────┘

►──┬─────────────────────────────────┬──┬────────────────────────────────┬──────────────────────►
   │                ┌─CLIENT─┐       │  └─parmform=──┬─OSLINK─────┬──────┘
   └─luwcontrol=────┴─SERVER─┴───────┘               ├─COMMDATA───┤
                                                     └─CICSOSLINK─┘

►──┬─────────────────────────────┬──┬──────────────────────────────────┬────────────────────────►
   └─remoteapptype=──┬─VG────┬───┘  │                ┌─GENERATION─┐    │
                     ├─NONVG─┤      └─remotebind=────┴─RUNTIME────┴────┘
                     └─ITF───┘

►──┬────────────────────────────────┬──────────────────────────────────────────────────────────►◄
   └─serverid=───server identifier──┘
```

The following attributes are ignored:
- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST contable=BINARY
```

DCE Protocol

Processing Flow for VisualAge Generator DCE Common Services Remote Call

This section shows the processing flow for a VisualAge Generator DCE common services remote call.

1. VisualAge Generator DCE Server is started using a configuration file which specifies the DCE principal name that the server will obtain its DCE authorizations from (equivalent to a DCE userid), the location and the serverid name for binding information advertising, the Access Control List (ACL) object for client authorization, and the server programs that the server is authorized to process. The server programs are specified in one of two different groups; those in which secure DCE (authenticated RPC) communications are required and those which can be accessed via unsecured (unauthenticated RPC) DCE communications.

   The server obtains an object UUID for each program from the CDS object /.:/Servers/VAGenerator/SERVERID/program. If the program object is not defined, one will be created for it. The server uses the program object UUIDs when it advertises its binding information.

2. VisualAge Generator DCE Server advertises each of the programs that it services. The Cell Directory Services (CDS) location used for advertising the binding information is /.:/Servers/VAGenerator/SERVERID/LOCATION.
3. VisualAge Generator client retrieves the object UUID for the program from /.:/Servers/VAGenerator/SERVERID/program-name. It then uses the object UUID to request the binding information for a server which services the program from the CDS location /.:/Servers/VAGenerator/SERVERID/LOCATION. If there are multiple server bindings that match the search criteria, DCE will randomly return one of them.
4. VisualAge Generator client will setup for authenticated RPC, if DCESECURE is specified in the linkage table.
5. VisualAge Generator client performs data conversion on passed parameters as specified in the `contable` attribute of the client linkage table (runtime or generation time as applicable).
6. VisualAge Generator client makes remote call to VisualAge Generator DCE Server.
7. VisualAge Generator DCE Server checks if VisualAge Generator client is authorized to use server program (via DCE CDS Access Control List) and if the appropriate level of communication security is being used from the client.
8. VisualAge Generator DCE Server checks if the server program requested is one that it is authorized to process (via initial configuration file).
9. VisualAge Generator DCE Server processes client request, closes Logical Unit of Work, and returns data to client.

User Authentication and Authorization

User authentication is performed on the client via the DCE security server using the client’s DCE login identifier. The VisualAge Generator DCE server checks whether the client is authorized to call the DCE server using DCE ACL security services. User authorization is performed via the DCE ACL security services. The VisualAge Generator DCE server is told at startup time the DCE object ACL to use for checking client authorization for running the called server programs. There is only one ACL used per VisualAge Generator DCE server; therefore, the authorization is at the VisualAge Generator DCE server level and not the called program level. If a called program requires a special ACL, then another VisualAge Generator DCE server will have to be created or started. The test ACL attribute determines whether or not the client is authorized to execute the server program (if the client has test privileges on the ACL object, then the client is authorized to execute all server programs provided by the server).

Controlling the Unit of Work

All calls via the DCE common services are server units of work. All resource changes are committed when the server returns to the calling program.

Data Format Conversion

Code format conversion is performed on the server call as specified on the `contable` attribute for the server entry in the linkage table. Data is converted based on the structure of the parameters specified on the server call in the client program.

Error Handling

The standard error handling procedures described in “Communication Error Handling” on page 22 are supported with VisualAge Generator DCE common services. EZERT8 is set to the decimal value of the client access service reason code. EZERT8 is only set when the REPLY option is coded on the call to the remote server program. If a visual link is used to make the call from a GUI program, the REPLY option is used on the call.

Any errors trapped by DCE are passed to the client program with a corresponding CSO error message. The error message contains an insert with the DCE mnemonic. A non-zero return code from the called program is passed back to the client program with a corresponding CSO error message. The error message contains an insert with the return code from the called program. VisualAge Generator return codes are documented in the help for the message.

All errors are traced to the CSO trace file on the client and server machines, as applicable.

Configuring an OS/2 Client for an OS/2 Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call server programs via the DCE common services:

```
►►—:calllink—applname=program name—linktype=—REMOTE———————————————————————►

►—remotecomtype=—┬—DCE———————┬——┬——————————————————————┬——————————————————►
                 └—DCESECURE—┘  └—parmform=—COMMDATA———┘
```

```
►──┬───────────────────────────────────────┬──┬──────────────────────────────┬──────►
   └─contable=─┬─conversion table name─┬─┘  └─location=─┬─EZELOC──────┬─┘
               ├─'*' ──────────────────┤                └─system name─┘
               ├─EZECONVT──────────────┤
               ├─NONE──────────────────┤
               └─BINARY────────────────┘

►──┬───────────────────────────────┬──┬───────────────────────────────────┬────────►◄
   │              ┌─GENERATION─┐   │  └─serverid=──server identifier──┘
   └─remotebind=──┴─RUNTIME────┴───┘
```

The following attributes are ignored:
- remoteapptype
- externalname
- library
- luwcontrol

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=DCE
   location=Server1 remotebind=RUNTIME serverid=Test
```

Configuring an OS/2 Client for a Windows NT Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call server programs via the DCE common services:

```
►►──:calllink──applname=program name──linktype=─REMOTE───────────────────────────────►

►─remotecomtype=──┬─DCE───────┬──┬──────────────────────┬────────────────────────────►
                  └─DCESECURE─┘  └─parmform=─COMMDATA───┘

►──┬───────────────────────────────────────┬──┬──────────────────────────────┬──────►
   └─contable=─┬─conversion table name─┬─┘  └─location=─┬─EZELOC──────┬─┘
               ├─'*' ──────────────────┤                └─system name─┘
               ├─EZECONVT──────────────┤
               ├─NONE──────────────────┤
               └─BINARY────────────────┘

►──┬───────────────────────────────┬──┬───────────────────────────────────┬────────►◄
   │              ┌─GENERATION─┐   │  └─serverid=──server identifier──┘
   └─remotebind=──┴─RUNTIME────┴───┘
```

The following attributes are ignored:

- remoteapptype
- externalname
- library
- luwcontrol

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=DCE
   location=Server1 remotebind=RUNTIME serverid=Test
```

Configuring an OS/2 Client for an AIX Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call server programs via the DCE common services:

```
►►──:calllink──applname=program name──linktype=─REMOTE──────────────────────────────►

►─remotecomtype=─┬─DCE───────┬──┬──────────────────────┬────────────────────────────►
                 └─DCESECURE─┘  └─parmform=─COMMDATA───┘

►─┬────────────────────────────────────────┬──┬──────────────────────────────┬──────►
  └─contable=─┬─conversion table name─┬────┘  └─location=─┬─EZELOC──────┬────┘
              ├─'*'───────────────────┤                   └─system name─┘
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬─────────────────────────────────┬──┬──────────────────────────────────┬───────►◄
  │               ┌─GENERATION─┐    │  └─serverid=──server identifier─────┘
  └─remotebind=───┴─RUNTIME────┴────┘
```

The following attributes are ignored:
- remoteapptype
- externalname
- library
- luwcontrol

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=DCE
   location=Server1 contable=BINARY remotebind=RUNTIME serverid=Test
```

DIRECT Protocol

Error Handling

The standard error handling procedures described in “Communication Error Handling” on page 22 are supported with IPC and DIRECT server calls. EZERT8 is set to the decimal value of the client access service reason code. The reason code is the same as the number portion of error message CSOnnnna as listed in the *VisualAge Generator Messages and Problem Determination Guide* document.

Configuring an OS/2 Client for an OS/2 Server

Advantages of IPC and DIRECT Protocols: The protocols IPC and DIRECT provide a method for client programs to call server programs where both reside on the same system. A local call from the client program to the server program might be used, but there are advantages to making this call remotely via the VisualAge Generator middleware in certain cases. Futhermore, these advantages differ for the provided protocols, IPC and DIRECT, and are described below.

Calls to local C++ applications for database access from Smalltalk GUI applications where the GUI application might run in the same session with other unrelated GUI applications that require database access: GUI applications begin VisualAge Generator communications sessions when calling server programs. When the IPC protocol is used, each call from a different GUI would be made under a different process, each of which can have its own database connection and unit of work. This would not be possible if the client program used a local call to access the server program.

If two GUI applications need to be in the same unit of work, start the second GUI from the first and define the second GUI with the VAGen inheritsCommSession attribute set to true in the AbtAppBldrPart or AbtAppBldrView class. This way, local servers called by both GUI applications will share the same communications session, database connection, and unit of work.

Calls to generated C++ application from Java clients where the C++ application is to run on a web server: Multiple calls might arrive simultaneously, thus the need for multiple, independent server-controlled units of work provided by the IPC protocol.

When using the DIRECT protocol, the call is similar to a local call directly from a client program to a server program except that the VisualAge Generator middleware performs the call. Because no separate process is created, the DIRECT protocol provides improved performance over the IPC protocol. The following are scenarios for using the DIRECT protocol:

Calls to a server program running under the control of VisualAge Generator test facility from Java, VisualBasic, or PowerBuilder, where the client and server resides on the same system.

Calls to a server program running under the control of Component Broker, which maintains the database connection itself. In such an environment, the server program should not run under a separate process so that the database connection cannot be inadvertently closed when the process ends.

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via DIRECT:

```
►►—:calllink—applname=program name—linktype=—CSOCALL—remotecomtype=—DIRECT————————►◄
```

While the specification of:

```
►►—contable=—┬—conversion table name—┬——————————————————————————————————————————————►◄
             ├—'*'———————————————————┤
             ├—EZECONVT——————————————┤
             ├—NONE——————————————————┤
             └—BINARY————————————————┘
```

is permissible, data conversion is generally not necessary when the client and server reside on the same system. All other linkage table entries are ignored on a local (DIRECT) call.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=CSOCALL remotecomtype=DIRECT
```

IPC Protocol

Advantages of IPC and DIRECT Protocols

The protocols IPC and DIRECT provide a method for client programs to call server programs where both reside on the same system. A local call from the client program to the server program might be used, but there are advantages to making this call remotely via the VisualAge Generator middleware in certain cases. Futhermore, these advantages differ for the provided protocols, IPC and DIRECT, and are described below.

Calls to local C++ applications for database access from Smalltalk GUI applications where the GUI application might run in the same session with other unrelated GUI applications that require database access: GUI applications begin VisualAge Generator communications sessions when calling server programs. When the IPC protocol is used, each call from a different GUI would be made under a different process, each of which can have its own database connection and unit of work. This would not be possible if the client program used a local call to access the server program.

If two GUI applications need to be in the same unit of work, start the second GUI from the first and define the second GUI with the VAGen inheritsCommSession attribute set to true in the AbtAppBldrPart or AbtAppBldrView class. This way, local servers called by both GUI applications will share the same communications session, database connection, and unit of work.

Calls to generated C++ application from Java clients where the C++ application is to run on a web server: Multiple calls might arrive simultaneously, thus the need for multiple, independent server-controlled units of work provided by the IPC protocol.

When using the DIRECT protocol, the call is similar to a local call directly from a client program to a server program except that the VisualAge Generator middleware performs the call. Because no separate process is created, the DIRECT protocol provides improved performance over the IPC protocol. The following are scenarios for using the DIRECT protocol:

Calls to a server program running under the control of VisualAge Generator test facility from Java, VisualBasic, or PowerBuilder, where the client and server resides on the same system.

Calls to a server program running under the control of Component Broker, which maintains the database connection itself. In such an environment, the server program should not run under a separate process so that the database connection cannot be inadvertently closed when the process ends.

Error Handling

The standard error handling procedures described in "Communication Error Handling" on page 22 are supported with IPC and DIRECT server calls. EZERT8 is set to the decimal value of the client access service reason code. The reason code is the same as the number portion of error message CSOnnnna as listed in the *VisualAge Generator Messages and Problem Determination Guide* document.

Configuring an OS/2 Client for an OS/2 Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via IPC:

```
►►—:calllink—applname=program name—linktype=—CSOCALL—remotecomtype=—IPC——————————►◄
```

While the specification of:

```
►►—contable=—┬conversion table name┬————————————————————————————————————————————►◄
             ├’*’ ——————————————————┤
             ├EZECONVT——————————————┤
             ├NONE——————————————————┤
             └BINARY————————————————┘
```

is permissable, data conversion is generally not necessary when the client and server reside on the same system. All other linkage table entries are ignored on a local (IPC) call.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=CSOCALL remotecomtype=IPC
```

LU2 Protocol

Customizing a Communication Client for LU2

LU 2 signing on and off of a CICS/ESA region is accomplished through the use of two script files, MFLOGON.SCR and MFLOGOFF.SCR. These script files emulate the commands a user would use to establish a LU 2 session, signing on to a CICS region and then stopping the session. A 24x80 screen size is required to use the VisualAge Generator LU 2 support.

MFLOGON.SCR
Establishes the session and optionally signs on to the region.

MFLOGOFF.SCR
Handles signing off and stopping the session.

The LU2_LOGON_SCRIPT and the LU2_LOGOFF_SCRIPT entries in the CSO.INI file identify the directory where the MFLOGON.SCR and MFLOGOFF.SCR script files are located.

Examples of the MFLOGON.SCR and MFLOGOFF.SCR script files are provided in the samples directory.

The following example shows the LU 2 statements in the CSO.INI file that identify the location of the logon and logoff scripts:

```
LU2_LOGON_SCRIPT=C:\CSODIR\MFLOGON.SCR
LU2_LOGOFF_SCRIPT=C:\CSODIR\MFLOGOFF.SCR
```

In the VisualAge Generator client/server communication script language, every non-blank line must begin with a comment marker (//) or a script verb. The script verbs are shown in Table 9:

Table 9. VisualAge Generator Client/Server Communication Script Verbs

Verb	Description
CONNECTPS	Establishes a connection between the client program and one of the supplied session IDs. • Syntax: CONNECTPS <a_list_of_session_IDs>; • Return: 0 if successful, nonzero otherwise **Example**: `CONNECTPS a b c;`
DISCONNECTPS	Drops the connection between the client program and the terminal. • Syntax: DISCONNECTPS; • Return: 0 if successful, nonzero otherwise **Example**: `DISCONNECTPS;`
SEND	Sends one or more keystrokes to the current connect terminal session. The following keystrokes are supported: • CLEAR • END • ENTER • HOME • PA1 through PA3 • PF1 through PF24 • SPACE • SYSREQ • TAB • Syntax: SEND <one_or_more_keystokes>; • Return: 0 if successful, nonzero otherwise **Example**: `SEND ENTER;`

Table 9. VisualAge Generator Client/Server Communication Script Verbs (continued)

Verb	Description
SETCURSOR	Sets the cursor position. • Syntax: SETCURSOR <row_number> <column_number>; • Return: None **Example**: `SETCURSOR 24 1;`
PAUSE	Pauses for the number of seconds specified. • Syntax: PAUSE <number_of_seconds>; • Return: None **Example**: `PAUSE 3;`
SEARCHPS	Searches the host presentation space for the specified string. The string search is case sensitive. • Syntax: SEARCHPS <a_string>; • Return: 0 if the string is found on the host screen, nonzero otherwise **Example**: `SEARCHPS WELCOME TO CICS/ESA;`
WRITE	Copies text directly onto the host screen at the current cursor position. • Syntax: WRITE <a_strings>; • Return: 0 if successful, nonzero otherwise **Example**: `WRITE CESN USERID=SYSAD,PS=SYSAD;`

At run time, several of the script verbs return a value to the VisualAge Generator client/server communication support services. When a script verb returns a nonzero value, the following happens:

- VisualAge Generator client/server communication support stops interpreting the script file.
- An error code is set for use by the client program.

Configuring an OS/2 Client for a CICS for MVS/ESA Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via LU 2:

```
►►──:calllink──applname=program name──linktype=─REMOTE──parmform=─COMMDATA──────────►

►─remotecomtype=─LU2──┬─────────────────────────────┬───────────────────────────────►◄
                      └─serverid=───server identifier──┘
```

Example Linkage Table

```
:calllink applname=ELACVP5 parmform=COMMDATA linktype=REMOTE remotecomtype=LU2
   location=CICSTST
```

TCP/IP Protocol

Creating a TCP/IP Services File Entry

The client machine must have an entry for the TCP/IP Serverid added to its TCP/IP services file. On OS/2, the services file resides in the etc subdirectory which can be located by issuing the command ″set etc″ (without the double quotes).

Error Handling

The standard error handling procedures described in “Communication Error Handling” on page 22 are supported with TCPIP server calls. EZERT8 is set to the decimal value of the client access service reason code. The reason code is the same as the number portion of error message CSOnnnna as listed in the *VisualAge Generator Messages and Problem Determination Guide* document.

Configuring an OS/2 Client for an OS/2 Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─TCPIP────────►

►─┬────────────────────────────────────────┬────────────────────────────────────────►
  └─contable=─┬─conversion table name─┬─┘
              ├─'*' ──────────────────┤
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘
```

```
►─┬──────────────────────────────────────────┬──────────────────────────────►
  └─location=──system name_tcpip_hostname────┘

►─┬──────────────────────────────────────┬──────────────────────────────────►◄
  └─serverid=──tcpip_server identifier───┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to “Conversion Table by Language and Platform” on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 remotebind=RUNTIME serverid=port1
```

Configuring an OS/2 Client for a Windows NTServer

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─TCPIP──────────►

►─┬────────────────────────────────────────┬─────────────────────────────────►
  └─contable=─┬─conversion table name──┬───┘
              ├─'*'────────────────────┤
              ├─EZECONVT───────────────┤
              ├─NONE───────────────────┤
              └─BINARY─────────────────┘

►─┬──────────────────────────────────────────┬──────────────────────────────►
  └─location=──system name_tcpip_hostname────┘

►─┬──────────────────────────────────────┬──────────────────────────────────►◄
  └─serverid=──tcpip_server identifier───┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to “Conversion Table by Language and Platform” on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 remotebind=RUNTIME serverid=port1
```

Configuring an OS/2 Client for an AIX Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─TCPIP──────────►

►─┬────────────────────────────────────────┬─────────────────────────────────────────►
  └─contable=─┬─conversion table name─┬─┘
              ├─'*' ──────────────────┤
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬─────────────────────────────────────────────┬────────────────────────────────────►
  └─location=───system name_tcpip_hostname───┘

►─┬─────────────────────────────────────────┬───────────────────────────────────────►◄
  └─serverid=───tcpip_server identifier───┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to “Conversion Table by Language and Platform” on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 contable=BINARY remotebind=RUNTIME serverid=port1
```

Configuring an OS/2 Client for an HP-UX Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─TCPIP──────────►

►──┬──────────────────────────────────────┬────────────────────────────────────────────►
   └─contable=─┬─conversion table name─┬──┘
               ├─'*' ──────────────────┤
               ├─EZECONVT──────────────┤
               ├─NONE──────────────────┤
               └─BINARY────────────────┘

►──┬──────────────────────────────────────────────┬────────────────────────────────────►
   └─location=───system name_tcpip_hostname───────┘

►──┬──────────────────────────────────────────┬────────────────────────────────────────►◄
   └─serverid=───tcpip_server identifier──────┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to "Conversion Table by Language and Platform" on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 contable=BINARY remotebind=RUNTIME serverid=port1
```

Configuring an OS/2 Client for a Solaris Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─TCPIP──────────►
```

```
►─┬────────────────────────────────────┬─────────────────────────────────────────────►
  └─contable=─┬─conversion table name─┬─┘
              ├─'*' ──────────────────┤
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬──────────────────────────────────────────┬───────────────────────────────────────►
  └─location=──system name_tcpip_hostname────┘

►─┬──────────────────────────────────────┬──────────────────────────────────────────►◄
  └─serverid=──tcpip_server identifier───┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to “Conversion Table by Language and Platform” on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 contable=BINARY remotebind=RUNTIME serverid=port1
```

Configuring an OS/2 Client for a VM/ESA Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─TCPIP──────────►

►─┬────────────────────────────────────┬─────────────────────────────────────────────►
  └─contable=─┬─conversion table name─┬─┘
              ├─'*' ──────────────────┤
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬──────────────────────────────────────────┬───────────────────────────────────────►
  └─location=──system name_tcpip_hostname────┘
```

```
►──┬─────────────────────────────────┬──────────────────────────────────────►◄
   └─serverid=──tcpip_server identifier──┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to "Conversion Table by Language and Platform" on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 contable=ELACNENU remotebind=RUNTIME serverid=port1
```

Configuring an OS/2 Server

Summary Table of Valid Clients and Protocols

Table 10. Valid Clients and Protocols for VisualAge Generator Servers on the OS/2 Platform

Server Platforms	Client Platforms				
	OS/2 (GUI, C++, ITF)	**Windows 95 and Windows 98 (GUI)**	**Windows NT (GUI, C++, ITF)**	**AIX (C++)**	**Solaris (C++)**
OS/2	TCP/IP, DCE, IPC, DIRECT	TCP/IP, DCE	TCP/IP, DCE	TCP/IP, DCE	TCP/IP

DCE Protocol

List of Valid Clients

- OS/2 (GUI, C++, ITF)
- Windows 95 and Windows 98 (GUI)
- Windows NT (GUI, C++, ITF)
- AIX (C++)

Identifying the Server Location

The server location is determined by the bindings advertised in the DCE CDS database. The bindings are found in the DCE CDS location object located at /.:/Servers/VAGenerator/SERVERID/LOCATION. Identify the server location to the client by specifying the location identifier in the `location` linkage table attribute in the entry for the server program, or set the location dynamically in the client program at runtime using EZELOC.

Linkage Table Attributes for Generating Server Programs

To generate native OS/2, AIX, or Windows NT server programs, no linkage table entry is required. If the server program is going to make remote calls (it is the second tier of a three tier client/server system) then a linkage table needs to be provided at generation time just as if it was a client program.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=DCE
   location=Server1 remotebind=GENERATION serverid=Test
```

DCE CDS Entries Required for VisualAge Generator DCE Servers

All VisualAge Generator DCE CDS entries are placed under the /.:/Servers/VAGenerator directory. Each serverid must have a directory, /.:/Servers/VAGenerator/serverid.

Note: It might be helpful to equate serverid with an application system and location with a server or group of servers that process requests for the application system.

DIRECT Protocol

List of Valid Clients

- OS/2 (GUI, C++, ITF)

Linkage Table Attributes for Generating Server Programs

To generate native OS/2 server programs no linkage table entry is required.

Identifying a C++ Server Location

When using the IPC or DIRECT protocols, server programs are located using the LIBPATH environment variable. You need to ensure that the server DLL directory is specified in the LIBPATH.

IPC Protocol

List of Valid Clients

- OS/2 (GUI, C++, ITF)

Linkage Table Attributes for Generating Server Programs

To generate native OS/2 server programs no linkage table entry is required.

Identifying the Server Location

When using the IPC or DIRECT protocols, server programs are located using the LIBPATH environment variable. You need to ensure that the server DLL directory is specified in the LIBPATH.

Configuring an OS/2 Server

TCP/IP Protocol

List of Valid Clients

- OS/2 (GUI, C++, ITF)
- Windows 95 and Windows 98 (GUI)
- Windows NT (GUI, C++, ITF)
- AIX (C++)
- Solaris (C++)

Linkage Table Attributes for Generating Server Programs

To generate native OS/2 server programs no linkage table entry is required.

Identifying a C++ Server Location

The TCP/IP support servers locate the server programs via the LIBPATH environment variable, so you need to ensure that the server DLL directory is specified in the LIBPATH.

C++ Server Program Set Up and Operation

The server uses a configuration file for specifying the TCP/IP service name to listen on. The configuration file is optional as there are predefined default values that will be used. The configuration file can also be used to modify a ″performance″ parameter, tcp_start_process.

Start the TCP/IP server by issuing the command:

```
CSOTCPS "config_filename"
```

The configuration file is located via the following search order:

1. File specified on the command line
2. The file named CSO.INI in the directory specified by the CSODIR environment variable.
3. The file named CSO.INI in the current directory

The search ends when the first one of the above conditions is met. Once a file is identified, the contents of the file are used if possible. If the file cannot be opened for any reason, a warning message is printed on the console and the default values are used. CSOTCPS will display the configuration filename and values being used prior to starting to listen for incoming requests. The default values are:

```
TCP/IP service name: VAGenerator
tcp_start_process: 4
```

Where:

TCP/IP service name	Specifies the service name that will be used to look up the TCP/IP port number that

	CSOTCPS will listen on for incoming requests. This entry is case sensitive and must match an entry in the TCP/IP services file. For Windows NT, AIX, and HP-UX the TCP/IP SERVICES file is used.
tcp_start_process	Specifies the number of server processes which will be prestarted. This number must be at least 1. Very lightly loaded systems could get away with specifying a lower value if you needed to minimize the number of running processes. Heavily loaded systems could see a moderate performance gain by increasing this value.

Sample TCP/IP Entries from CSO.INI File: The following TCP/IP entries are located in the sample CSO.INI file. All entries must start in column 1. Comments are allowed within the file by placing a semicolon (;) in column 1 of the comment line. You only need to specify an entry if you wish to override the default value. If duplicate entries are specified in the file, the last entry in the file is used (just like environment variables in the config.sys file).

```
tcp_service_name=VAGenerator
tcp_start_process=2
```

Chapter 6. CICS for OS/2 Platform

How to use this chapter

This chapter contains platform and protocol information necessary to configure a client program or a server program. The chapter is divided into configuration sections: client programs and server programs. Each configuration section is organized by protocol. Each protocol section contains information organized by target platforms.

Use the following method to locate the information you need:

1. Verify that this chapter is appropriate to your runtime environment.

   Ensure that the runtime environment of your program matches the platform on which this chapter is based. Turn to the appropriate chapter if the runtime environment of your program does not match the platform named in the title of this chapter.
2. Determine which type of program you want to configure: client program or server program.
3. Determine which configuration section in the chapter is appropriate for your program.
4. Find the intended target platform of the program you are configuring in the Summary Table of Valid Clients and Protocols or Summary Table of Valid Servers and Protocols.

   If you are configuring a client program, find the target platform of the intended server program in the column of valid platforms.

   If you are configuring a server program, find the target platform of the intended client program in the row of valid platforms.
5. Chose a protocol based on the information in the table.

   If you are configuring a client program, find the cell in the table where the row of the intended target platform intersects the column of the client platform. Select a protocol from the list of available protocols for this client/server combination.

   If you are configuring a client program, find the cell in the table where the column of the intended target platform intersects the row of the server platform. Select a protocol from the list of available protocols for this client/server combination.
6. Determine the name of the appropriate protocol section within the configuration section.

Turn to the part of the configuration section indicated by the protocol name. Here you will find information common to all target platforms valid for this protocol.

7. Determine the name of the appropriate target platform section within the protocol section.

   You will find platform specific information for the chosen protocol in the section named for your intended target platform.

Example

```
Chapter: CICS for OS/2 Platform
Type of program to configure: Client program
Configuration section: Configuring an CICS for OS/2 client
Intended target platform: CICS for MVS/ESA
(i.e. a CICS for MVS/ESA server program)
Chosen protocol: CICS DPL
Protocol section:  CICS DPL Protocol
Target platform section: Configuring a CICS for OS/2 Client for a
CICS for MVS/ESA Server
```

You will encounter syntax diagrams in this chapter. The diagrams are used to show programming syntax as follows:

```
►►──:RequiredTerm──RequiredTerm=RequiredValue──RequiredTerm=─┬─OptionalValue─┬──────────►
                                                             └─OptionalValue─┘

►──┬──────────────────────────────────────┬──┬────────────────────────────────────┬─────►
   │                  ┌─DefaultValue──┐   │  └─OptionalTerm=RequiredValue─┘
   └─OptionalTerm=────┴─OptionalValue─┴───┘

►──┬──────────────────────────────────────┬─────────────────────────────────────────────►◄
   └─OptionalTerm=──┬─OptionalValue─┬─────┘
                    └─OptionalValue─┘
```

RequiredTerm	A required term is a term that must be specified. Required terms are depicted in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required terms.
RequiredValue	A required value is a value that must be specified for a term. A required value is shown in italics in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required values.
OptionalTerm	An optional term is a term that can be

	specified. An optional term is depicted in syntax diagrams in an alternate path from the preceding term.
OptionalValue	An optional value is a value that can be specified for a term. An optional value is shown in italics in syntax diagrams in an alternate path from the preceding term along with other optional values or a default value.
DefaultValue	A default value is the value that is specified by default for a term. A default value is shown in italics in syntax diagrams in an alternate path above other optional values.

Configuring a CICS for OS/2 Client

Summary Table of Valid Servers and Protocols

Table 11. Valid Servers and Protocols for VisualAge Generator Clients on the CICS for OS/2 Platform

Server Platforms	Protocols for CICS for OS/2 Client Platform
CICS for AIX	CICS DPL
CICS for MVS/ESA	CICS DPL
CICS for OS/2	CICS DPL
CICS for Solaris	CICS DPL
CICS for VSE/ESA	CICS DPL
CICS for Windows NT	CICS DPL

CICS DPL Protocol

User Authentication
The user associated with the CICS client transaction must be authorized to run the client transaction and the server transaction. The user ID and password are identified by the CICS transaction manager on the client system.

Setting Up Communication Links
The communication link between the client and server systems must be defined to the CICS systems to allow a distributed program link from one system to the other. Refer to the CICS intercommunication manual for your CICS systems for more information.

Controlling the Unit of Work
Use the `luwcontrol` linkage table attribute to indicate whether the server updates are automatically committed on return or the client controls the unit of work. If the client is controlling the unit of work, subsequent server calls

must go to the same system and transaction under client-controlled unit of work until a commit or roll back is requested. The server cannot issue EZECOMIT or EZEROLLB calls if client-controlled unit of work is specified.

Data Format Conversion

Code format conversion is performed on the server call as specified in the `contable` linkage attribute table and in EZECONVT. Code is converted based on the structure of the arguments that are specified on the CALL statement in the client program.

Error Handling

The standard error handling procedures described below are supported with CICS DPL. If the DPL is not successful for any reason, including unavailability of the communication link, error information is returned with CICS in the CICS EIB (EXEC interface block).

If the REPLY option is not specified on the CALL statement, the calling program ends with error messages. If REPLY is specified, no messages are written or logged, and EZERT8 is set to one of the following values:

Value	Meaning
00000000	Successful call and return
00000204	Program name not valid
00000207	System identifier not valid
00000208	Link out of service
ffrrrrrr	Other CICS error where ff is the hexadecimal representation of EIBFN byte 0, and rrrrrr is the hexadecimal representation of EIBRCODE bytes 0–2.

Configuring a CICS for OS/2 Client for a CICS for OS/2 Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►─:calllink─applname=program name─linktype=─REMOTE─┬────────────────────┬───────►
                                                    └─parmform=─COMMDATA─┘

►─┬──────────────────────────────────────────┬─────────────────────────────────────►
  │              ┌─GENERATION─┐               │
  └─remotebind=──┴────────────┴──RUNTIME─────┘
```

```
►──┬──────────────────────────────────────┬──┬──────────────────────────────┬──────►
   └─contable=──┬─conversion table name─┬─┘  └─location=──┬─EZELOC──────┬─┘
                ├─'*'───────────────────┤                 └─system name─┘
                ├─EZECONVT──────────────┤
                ├─NONE──────────────────┤
                └─BINARY────────────────┘

►──┬──────────────────────────────┬──┬──────────────────────────────┬──────────────►
   │              ┌─CLIENT─┐      │  └─remoteapptype=──┬─VG────┬──┘
   └─luwcontrol=──┴─SERVER─┴──────┘                    ├─NONVG─┤
                                                       └─ITF───┘

►──┬─────────────────────────────────────┬────────────────────────────────────────►◄
   └─serverid=──server identifier────────┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CS07 LUWCONTROL=CLIENT
```

Configuring a CICS for OS/2 Client for a CICS for Windows NT Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►──:calllink──applname=program name──linktype=─REMOTE──┬─────────────────────┬────►
                                                        └─parmform=─COMMDATA──┘

►──┬──────────────────────────────────────────────┬───────────────────────────────►
   │              ┌─GENERATION─┐                  │
   └─remotebind=──┴────────────┴──RUNTIME─────────┘

►──┬──────────────────────────────────────┬──┬──────────────────────────────┬──────►
   └─contable=──┬─conversion table name─┬─┘  └─location=──┬─EZELOC──────┬─┘
                ├─'*'───────────────────┤                 └─system name─┘
                ├─EZECONVT──────────────┤
                ├─NONE──────────────────┤
                └─BINARY────────────────┘
```

```
►─┬──────────────────────────────┬─┬───────────────────────────────┬──────────────────────►
  │              ┌─CLIENT─┐      │ └─remoteapptype=─┬─VG────┬─┘
  └─luwcontrol=──┴─SERVER─┴──────┘                  ├─NONVG─┤
                                                    └─ITF───┘

►─┬─────────────────────────────────────┬──────────────────────────────────────────────►◄
  └─serverid=──server identifier────────┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CS07 LUWCONTROL=CLIENT
```

Configuring a CICS for OS/2 Client for a CICS for AIX Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►──:calllink──applname=program name──linktype=─REMOTE──┬──────────────────────┬────────►
                                                        └─parmform=─COMMDATA───┘

►─┬─────────────────────────────────────────────┬──────────────────────────────────────►
  │               ┌─GENERATION─┐                │
  └─remotebind=───┴────────────┴──RUNTIME───────┘

►─┬──────────────────────────────────────────────┬─┬──────────────────────────────┬────►
  └─contable=─┬─conversion table name──────────┬─┘ └─location=─┬─EZELOC──────┬────┘
              ├─'*'────────────────────────────┤               └─system name─┘
              ├─EZECONVT───────────────────────┤
              ├─NONE───────────────────────────┤
              └─BINARY─────────────────────────┘

►─┬──────────────────────────────┬─┬───────────────────────────────┬──────────────────────►
  │              ┌─CLIENT─┐      │ └─remoteapptype=─┬─VG────┬─┘
  └─luwcontrol=──┴─SERVER─┴──────┘                  ├─NONVG─┤
                                                    └─ITF───┘
```

```
►─┬────────────────────────────────────┬──────────────────────────────────────────────►◄
  └─serverid=──server identifier───────┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CSO7 contable=BINARY LUWCONTROL=CLIENT
```

Configuring a CICS for OS/2 Client for a CICS for MVS/ESA Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►──:calllink──applname=program name──linktype=─REMOTE──┬──────────────────────┬──────►
                                                        └─parmform=─COMMDATA───┘

►─┬──────────────────────────────────────┬────────────────────────────────────────────►
  │                ┌─GENERATION─┐        │
  └─remotebind=────┴────────────┴─RUNTIME┘

►─┬───────────────────────────────────────┬─┬────────────────────────────┬────────────►
  └─contable=──┬─conversion table name─┬──┘ └─location=──┬─EZELOC──────┬─┘
               ├─'*'───────────────────┤                 └─system name─┘
               ├─EZECONVT──────────────┤
               ├─NONE──────────────────┤
               └─BINARY────────────────┘

►─┬──────────────────────────────┬─┬──────────────────────────────┬───────────────────►
  │                ┌─CLIENT─┐    │ └─remoteapptype=──┬─VG────┬────┘
  └─luwcontrol=────┴─SERVER─┴────┘                   ├─NONVG─┤
                                                     └─ITF───┘

►─┬────────────────────────────────────┬──────────────────────────────────────────────►◄
  └─serverid=──server identifier───────┘
```

The following attributes are ignored:
- externalname
- library

- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CS07 contable=ELACNENU LUWCONTROL=CLIENT
```

Configuring a CICS for OS/2 Client for a CICS for VSE/ESA Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►─:calllink─applname=program name─linktype=─REMOTE─┬──────────────────────┬─────────►
                                                    └─parmform=─COMMDATA───┘

►─┬────────────────────────────────────────┬───────────────────────────────────────────►
  │              ┌─GENERATION─┐            │
  └─remotebind=──┴────────────┴──RUNTIME───┘

►─┬─────────────────────────────────────┬──┬───────────────────────────────┬───────────►
  └─contable=──┬─conversion table name─┬┘  └─location=──┬─EZELOC──────┬───┘
               ├─'*'───────────────────┤                └─system name─┘
               ├─EZECONVT──────────────┤
               ├─NONE──────────────────┤
               └─BINARY────────────────┘

►─┬──────────────────────────────┬──┬──────────────────────────────┬───────────────────►
  │              ┌─CLIENT──┐     │  └─remoteapptype=──┬─VG────┬────┘
  └─luwcontrol=──┴─SERVER──┴─────┘                    ├─NONVG─┤
                                                      └─ITF───┘

►─┬─────────────────────────────────────┬──────────────────────────────────────────────►◄
  └─serverid=───server identifier───────┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CS07 contable=ELACNENU LUWCONTROL=CLIENT
```

Configuring a CICS for OS/2 Client for a CICS for Solaris Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►──:calllink──applname=program name──linktype=─REMOTE──┬───────────────────┬──────►
                                                         └─parmform=─COMMDATA─┘

►──┬──────────────────────────────────────────┬──────────────────────────────────────►
   │               ┌─GENERATION─┐             │
   └─remotebind=───┴────────────┴──RUNTIME────┘

►──┬────────────────────────────────────────┬─┬──────────────────────────────┬──────►
   └─contable=──┬─conversion table name─┬───┘ └─location=──┬─EZELOC──────┬──┘
                ├─'*'───────────────────┤                  └─system name─┘
                ├─EZECONVT──────────────┤
                ├─NONE──────────────────┤
                └─BINARY────────────────┘

►──┬─────────────────────────────┬─┬─────────────────────────────┬──────────────────►
   │              ┌─CLIENT─┐     │ └─remoteapptype=──┬─VG────┬───┘
   └─luwcontrol=──┴─SERVER─┴─────┘                   ├─NONVG─┤
                                                     └─ITF───┘

►──┬───────────────────────────────────┬──────────────────────────────────────────►◄
   └─serverid=───server identifier─────┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CS07 contable=BINARY LUWCONTROL=CLIENT
```

Configuring a CICS for OS/2 Server

Summary Table of Valid Clients and Protocols

Table 12. Valid Clients and Protocols for VisualAge Generator Servers on the CICS for OS/2 Platform

Server Platforms	**Client Platforms**					
	OS/2 (GUI, C++, ITF)	**Windows 95 and Windows 98 (GUI)**	**Windows NT (GUI, C++, ITF)**	**AIX (C++)**	**CICS**	**Solaris (C++)**
CICS for OS/2	CICS Client	CICS Client	CICS Client	CICS Client	CICS DPL	CICS Client

Notes:
CICS as a Client Platform refers to:
- CICS for AIX
- CICS for MVS/ESA
- CICS for OS/2
- CICS for Windows NT
- CICS for VSE/ESA
- CICS for Solaris

CICS Client Protocol

List of Valid Clients
- OS/2 (GUI, C++, ITF)
- Windows 95 and Windows 98 (GUI)
- Windows NT (GUI, C++, ITF)
- AIX (C++)
- Solaris (C++)

Identifying the Server Location
Specify the server location (system identifier) in the LOCATION linkage table attribute. You can also set the location dynamically in the client program at run time using EZELOC. If the server location is not specified, the default is the first entry in the CICS Client initialization file.

Linkage Table Attributes for Generating CICS Server Programs
Specify the following calllink attributes when you generate CICS server programs:

```
►►──:calllink──applname=program name──linktype=─REMOTE──┬───────────────────┬────────►◄
                                                        └─parmform=─COMMDATA─┘
```

The following attributes are ignored:
- contable
- externalname

- library
- location
- luwcontrol
- remoteapptype
- remotebind
- remotecomtype
- serverid

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA
```

Considerations for Defining the Server Program

If the remote program performs printing, it must move the print destination into EZEDESTP.

Server Program Set Up

The server program is generated and prepared as is any other CICS program on the server system. The server transaction must be defined to CICS on the server system. The default transaction name associated with a server program is the CICS-supplied mirror transaction, CPMI.

Specifying Parameter Format

Parameter format (the `parmform` attribute on the calllink tag) can be specified as COMMDATA (data values passed in the COMMAREA) or COMMPTR (pointers passed in the COMMAREA).

Using COMMPTR Format: Use COMMPTR for calling COBOL subroutines that were developed to use the non-VAGen parameter passing convention that uses pointers in the CICS COMMAREA. When the pointer format is used, the generated client program calls an intermediate catcher program (ELACSV, provided with VisualAge Generator host services) via the ECI instead of calling the server program directly. The catcher program receives the parameter data from the client, builds a pointer list in the COMMAREA that points to the individual parameters, and links to the server program. A program processing table (PPT) must be defined for the ELACSV program on the CICS server system.

COMMPTR format is only supported for MVS CICS, CICS for VSE/ESA, and CICS OS/2 servers.

Note: When calling COMMPTR programs in CICS for OS/2, you must specify `contable=NONE` in the linkage table entry.

Using COMMDATA Format: Use COMMDATA for calling generated programs and for calling COBOL subroutines where possible. COMMDATA provides better performance in calling remote programs. With COMMDATA, the server program is called directly without going through the catcher program.

CICS DPL Protocol

List of Valid Clients

- CICS for AIX
- CICS for MVS/ESA
- CICS for OS/2
- CICS for Windows NT
- CICS for VSE/ESA
- CICS for Solaris

Identifying the Server Location

You can specify the server location (CICS system identifier) on the `location` linkage table attribute, or dynamically set the location in the client program at runtime using EZELOC. If the location is not specified, the default location is the system identifier associated with the server program in the CICS program definition on the client system.

Linkage Table Attributes for Generating CICS Server Programs

Specify the following calllink attributes when you generate CICS server programs:

```
►►──:calllink──applname=program name──linktype=─REMOTE──┬────────────────────┬──────►◄
                                                        └─parmform=─COMMDATA─┘
```

The following attributes are ignored:

- contable
- externalname
- library
- location
- luwcontrol
- remoteapptype
- remotebind
- remotecomtype
- serverid

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA
```

Considerations for Defining the Server Program

If the remote program performs printing, it must move the print destination into EZEDESTP.

Server Program Set Up

The server program is generated and prepared as is any other CICS program on the server system. In addition, there must be a CICS program definition for the server program on the CICS client system containing the following information:

- Resident option = remote
- Remote system identifier
- Remote program name (optional)
- Remote transaction identifier (optional)

The server transaction must be defined to CICS on the server system. The default transaction name associated with a server program is the CICS-supplied mirror transaction, CPMI.

Chapter 7. OS/400 Platform

How to use this chapter

This chapter contains platform and protocol information necessary to configure a client program or a server program. The chapter is divided into configuration sections: client programs and server programs. Each configuration section is organized by protocol. Each protocol section contains information organized by target platforms.

Use the following method to locate the information you need:

1. Verify that this chapter is appropriate to your runtime environment.

   Ensure that the runtime environment of your program matches the platform on which this chapter is based. Turn to the appropriate chapter if the runtime environment of your program does not match the platform named in the title of this chapter.
2. Determine which type of program you want to configure: client program or server program.
3. Determine which configuration section in the chapter is appropriate for your program.
4. Find the intended target platform of the program you are configuring in the Summary Table of Valid Clients and Protocols or Summary Table of Valid Servers and Protocols.

   If you are configuring a client program, find the target platform of the intended server program in the column of valid platforms.

   If you are configuring a server program, find the target platform of the intended client program in the row of valid platforms.
5. Chose a protocol based on the information in the table.

   If you are configuring a client program, find the cell in the table where the row of the intended target platform intersects the column of the client platform. Select a protocol from the list of available protocols for this client/server combination.

   If you are configuring a client program, find the cell in the table where the column of the intended target platform intersects the row of the server platform. Select a protocol from the list of available protocols for this client/server combination.
6. Determine the name of the appropriate protocol section within the configuration section.

Turn to the part of the configuration section indicated by the protocol name. Here you will find information common to all target platforms valid for this protocol.

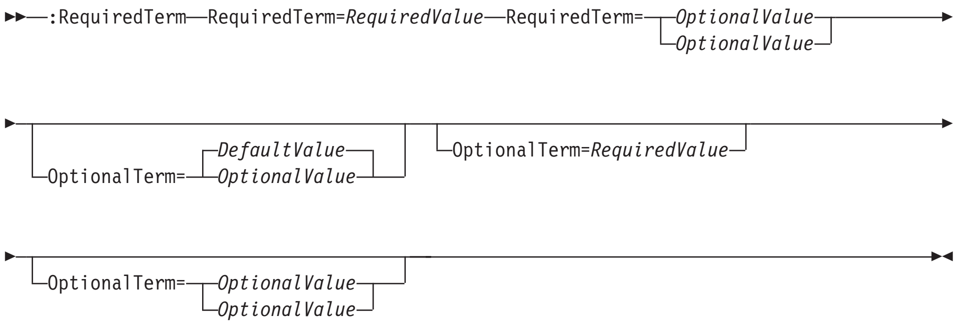
7. Determine the name of the appropriate target platform section within the protocol section.

   You will find platform specific information for the chosen protocol in the section named for your intended target platform.

Example

```
Chapter: OS/400 Platform
Type of program to configure: Server program
Configuration section: Configuring an OS/400 server
Intended target platform: OS/2 (i.e. an OS/2 client program)
Chosen protocol: Client Access/400
Protocol section: Client Access/400 Protocol
```

You will encounter syntax diagrams in this chapter. The diagrams are used to show programming syntax as follows:



RequiredTerm	A required term is a term that must be specified. Required terms are depicted in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required terms.
RequiredValue	A required value is a value that must be specified for a term. A required value is shown in italics in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required values.
OptionalTerm	An optional term is a term that can be

	specified. An optional term is depicted in syntax diagrams in an alternate path from the preceding term.
OptionalValue	An optional value is a value that can be specified for a term. An optional value is shown in italics in syntax diagrams in an alternate path from the preceding term along with other optional values or a default value.
DefaultValue	A default value is the value that is specified by default for a term. A default value is shown in italics in syntax diagrams in an alternate path above other optional values.

Configuring an OS/400 Client

Summary Table of Valid Servers and Protocols

Table 13. Valid Servers and Protocols for VisualAge Generator Clients on the OS/400 Platform

Server Platforms	Protocols for OS/400 Client Platform
AIX	N/A
CICS for AIX	N/A
HP-UX	N/A
IMS	N/A
CICS for MVS/ESA	N/A
OS/2	N/A
CICS for OS/2	N/A
OS/400	N/A
Solaris	N/A
CICS for Solaris	N/A
VM/ESA	N/A
CICS for VSE/ESA	N/A
Windows NT	N/A
Windows NT (Java)	N/A
CICS for Windows NT	N/A
Notes: N/A = Not available	

Configuring an OS/400 Server

Summary Table of Valid Clients and Protocols

Table 14. Valid Clients and Protocols for VisualAge Generator Servers on the OS/400 Platform

Server Platform	**Client Platforms**		
	OS/2 (GUI, C++, ITF)	**Windows 95 and Windows 98 (GUI)**	**Windows NT (GUI, C++, ITF)**
OS/400	CA/400	CA/400	CA/400

Client Access/400 Protocol

List of Valid Clients

- OS/2 (GUI, C++, ITF)
- Windows 95 and Windows 98 (GUI)
- Windows NT (GUI, C++, ITF)

Server Program Set Up

VisualAge Generator Server Programs: The server program is generated and prepared like any other OS/400 program, with the following additions:

- The program must be in the QVGN activation group. This is done for you during preparation unless you have modified the preparation templates.
- A runtime CL file is generated for the program, uploaded to the OS/400, and compiled into the same library into which the server program resides (DESTLIB generation option). The CL file name is applname_R (the program name with an _R appended).

  At run time, the client calls a catcher program, QVGNSRVR, instead of calling the server program directly. The catcher program calls the CL file before calling the server program. The CL file identifies the libraries that contain the server program and data files. It also sets the commitment control environment for the job if this is the first server call in the job. All server calls from the same client program to the same AS/400 system are included in the same job.

  You can customize the template for the CL file by doing the following:
  - Add any additional libraries required for database access or for programs or programs that the server program calls.
  - Modify the STRCMTCTL command to change the lock level (LCKLVL) or add additional keywords to the command.

  Leave the commit scope (CMTSCOPE) specified as *JOB in the CL file.

Non-VisualAge Generator Server Programs: A non-VisualAge Generator server program is prepared like other OS/400 programs, with the following additions:

- The program must be in the QVGN activation group. Following is an example of how to do this:

```
CRTPGM    PGM(MYLIBRARY/NONVGAPPL)         +
MODULE(*PGM)                      +
ENTMOD(*FIRST)                    +
BNDDIR(MYLIBRARY/MYBNDDIR)        +
ACTGRP(QVGN)                      +
USRPRF(*USER)                     +
AUT(*LIBCRTAUT)                   +
TEXT('Non-VisualAge Generator program')
```

- At run time, the client calls a catcher program, QVGNSRVR, instead of calling the server program directly. If the server program is a non-VisualAge Generator program, the catcher program calls the command language file QVGNRNCL before calling the server program.

  The sample directory that is shipped with VisualAge Generator Developer contains a sample file, QVGNRNCL.CLR. The sample file sets the job commitment control environment if this is the first server program or program call in the job.

  Ensure that a copy of QVGNRNCL is copied to the AS/400 and compiled into each library that contains a non-VisualAge Generator program called by a VisualAge Generator client program.

  You can customize the CL file by doing the following:
  - Add additional libraries required for file or database access or for programs or programs that the server programs call.
  - Modify the STRCMTCTL command to change the lock level (LCKLVL) or add additional keywords to the command.

  Leave the commit scope (CMTSCOPE) specified as *JOB in the CL file.

Chapter 8. HP-UX Platform

How to use this chapter

This chapter contains platform and protocol information necessary to configure a client program or a server program. The chapter is divided into configuration sections: client programs and server programs. Each configuration section is organized by protocol. Each protocol section contains information organized by target platforms.

Use the following method to locate the information you need:

1. Verify that this chapter is appropriate to your runtime environment.

   Ensure that the runtime environment of your program matches the platform on which this chapter is based. Turn to the appropriate chapter if the runtime environment of your program does not match the platform named in the title of this chapter.
2. Determine which type of program you want to configure: client program or server program.
3. Determine which configuration section in the chapter is appropriate for your program.
4. Find the intended target platform of the program you are configuring in the Summary Table of Valid Clients and Protocols or Summary Table of Valid Servers and Protocols.

   If you are configuring a client program, find the target platform of the intended server program in the column of valid platforms.

   If you are configuring a server program, find the target platform of the intended client program in the row of valid platforms.
5. Chose a protocol based on the information in the table.

   If you are configuring a client program, find the cell in the table where the row of the intended target platform intersects the column of the client platform. Select a protocol from the list of available protocols for this client/server combination.

   If you are configuring a client program, find the cell in the table where the column of the intended target platform intersects the row of the server platform. Select a protocol from the list of available protocols for this client/server combination.
6. Determine the name of the appropriate protocol section within the configuration section.

Turn to the part of the configuration section indicated by the protocol name. Here you will find information common to all target platforms valid for this protocol.

7. Determine the name of the appropriate target platform section within the protocol section.

   You will find platform specific information for the chosen protocol in the section named for your intended target platform.

Example

```
Chapter: HP-UX Platform
Type of program to configure: Server program
Configuration section: Configuring an HP-UX server
Intended target platform: OS/2 (i.e. an OS/2 client program)
Chosen protocol: TCP/IP
Protocol section:  TCP/IP Protocol
```

You will encounter syntax diagrams in this chapter. The diagrams are used to show programming syntax as follows:

```
►►─:RequiredTerm─RequiredTerm=RequiredValue─RequiredTerm=─┬─OptionalValue─┬──────────►
                                                          └─OptionalValue─┘


►─┬──────────────────────────────────────┬─┬──────────────────────────────┬──────────►
  │                 ┌─DefaultValue──┐    │ └─OptionalTerm=RequiredValue─┘
  └─OptionalTerm=───┴─OptionalValue─┴────┘


►─┬──────────────────────────────────────┬───────────────────────────────────────────►◄
  └─OptionalTerm=──┬─OptionalValue─┬─────┘
                   └─OptionalValue─┘
```

RequiredTerm	A required term is a term that must be specified. Required terms are depicted in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required terms.
RequiredValue	A required value is a value that must be specified for a term. A required value is shown in italics in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required values.
OptionalTerm	An optional term is a term that can be

	specified. An optional term is depicted in syntax diagrams in an alternate path from the preceding term.
OptionalValue	An optional value is a value that can be specified for a term. An optional value is shown in italics in syntax diagrams in an alternate path from the preceding term along with other optional values or a default value.
DefaultValue	A default value is the value that is specified by default for a term. A default value is shown in italics in syntax diagrams in an alternate path above other optional values.

Configuring an HP-UX Client

Summary Table of Valid Servers and Protocols

Table 15. Valid Servers and Protocols for VisualAge Generator Clients on the HP-UX Platform

Server Platforms	Protocols for HP-UX Client Platform
AIX	N/A
CICS for AIX	N/A
HP-UX	N/A
IMS	N/A
CICS for MVS/ESA	N/A
OS/2	N/A
CICS for OS/2	N/A
OS/400	N/A
Solaris	N/A
CICS for Solaris	N/A
VM/ESA	N/A
CICS for VSE/ESA	N/A
Windows NT (C++)	N/A
Windows NT (Java)	N/A
CICS for Windows NT	N/A
Notes: N/A = Not available	

Configuring an HP-UX Server

Summary Table of Valid Clients and Protocols

Table 16. Valid Clients and Protocols for VisualAge Generator Servers on the HP-UX Platform

Server Platform	**Client Platforms**				
	OS/2 (GUI, C++, ITF)	**Windows 95 and Windows 98 (GUI)**	**Windows NT (GUI, C++, ITF)**	**AIX (C++)**	**Solaris (C++)**
HP-UX	TCP/IP	TCP/IP	TCP/IP	TCP/IP	TCP/IP

TCP/IP Protocol

List of Valid Clients

- OS/2 (GUI, C++, ITF)
- Windows 95 and Windows 98 (GUI)
- Windows NT (GUI, C++, ITF)
- AIX (C++)
- Solaris (C++)

Identifying a C++ Server Location

The TCP/IP support servers locate the server programs via the LIBPATH environment variable, so you need to ensure that the server DLL directory is specified in the LIBPATH.

Linkage Table Attributes for Generating Server Programs

A linkage table is not required for HP-UX server programs.

Server Program Set Up and Operation

The server uses a configuration file for specifying the TCP/IP service name to listen on. The configuration file is optional as there are predefined default values that will be used.

Start the TCP/IP server by issuing the command:

```
CSOTCPS "config_filename"
```

The configuration file is located via the following search order:

1. File specified on the command line
2. The file named CSO.INI in the directory specified by the CSODIR environment variable.
3. The file named CSO.INI in the current directory

The search ends when the first one of the above conditions is met. Once a file is identified, the contents of the file are used if possible. If the file cannot be opened for any reason, a warning message is printed on the console and the default values are used. CSOTCPS will display the configuration filename and values being used prior to starting to listen for incoming requests. The default values are:

```
TCP/IP service name: VAGenerator
```

Where:

TCP/IP service name	Specifies the service name that will be used to look up the TCP/IP port number that CSOTCPS will listen on for incoming requests. This entry is case sensitive and must match an entry in the TCP/IP services file. For Windows NT, AIX, and HP-UX the TCP/IP SERVICES file is used.

Sample TCP/IP Entries from CSO.INI File: The following TCP/IP entries are located in the sample CSO.INI file. All entries must start in column 1. Comments are allowed within the file by placing a semicolon (;) in column 1 of the comment line. You only need to specify an entry if you wish to override the default value. If duplicate entries are specified in the file, the last entry in the file is used (just like environment variables in the config.sys file).

```
tcp_service_name=VAGenerator
```

Chapter 9. IMS Platform

How to use this chapter

This chapter contains platform and protocol information necessary to configure a client program or a server program. The chapter is divided into configuration sections: client programs and server programs. Each configuration section is organized by protocol. Each protocol section contains information organized by target platforms.

Use the following method to locate the information you need:

1. Verify that this chapter is appropriate to your runtime environment.

   Ensure that the runtime environment of your program matches the platform on which this chapter is based. Turn to the appropriate chapter if the runtime environment of your program does not match the platform named in the title of this chapter.
2. Determine which type of program you want to configure: client program or server program.
3. Determine which configuration section in the chapter is appropriate for your program.
4. Find the intended target platform of the program you are configuring in the Summary Table of Valid Clients and Protocols or Summary Table of Valid Servers and Protocols.

   If you are configuring a client program, find the target platform of the intended server program in the column of valid platforms.

   If you are configuring a server program, find the target platform of the intended client program in the row of valid platforms.
5. Chose a protocol based on the information in the table.

   If you are configuring a client program, find the cell in the table where the row of the intended target platform intersects the column of the client platform. Select a protocol from the list of available protocols for this client/server combination.

   If you are configuring a client program, find the cell in the table where the column of the intended target platform intersects the row of the server platform. Select a protocol from the list of available protocols for this client/server combination.
6. Determine the name of the appropriate protocol section within the configuration section.

Turn to the part of the configuration section indicated by the protocol name. Here you will find information common to all target platforms valid for this protocol.

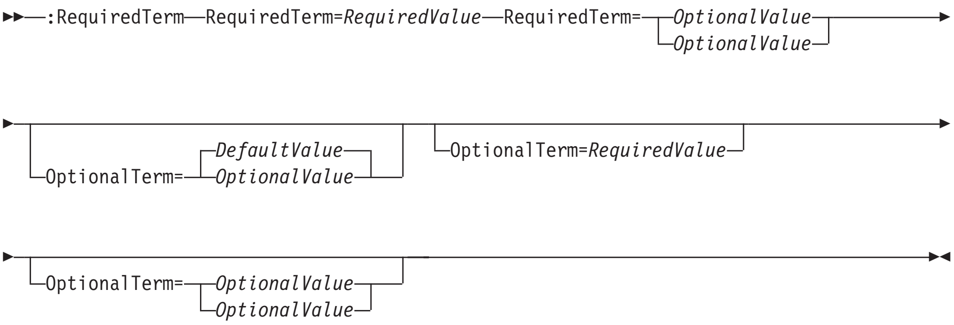
7. Determine the name of the appropriate target platform section within the protocol section.

   You will find platform specific information for the chosen protocol in the section named for your intended target platform.

Example

```
Chapter: IMS Platform
Type of program to configure: Server program
Configuration section: Configuring an IMS server
Intended target platform: OS/2 (i.e. an OS/2 client program)
Chosen protocol: APPC/IMS
Protocol section: APPC/IMS Protocol
```

You will encounter syntax diagrams in this chapter. The diagrams are used to show programming syntax as follows:



RequiredTerm	A required term is a term that must be specified. Required terms are depicted in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required terms.
RequiredValue	A required value is a value that must be specified for a term. A required value is shown in italics in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required values.
OptionalTerm	An optional term is a term that can be

	specified. An optional term is depicted in syntax diagrams in an alternate path from the preceding term.
OptionalValue	An optional value is a value that can be specified for a term. An optional value is shown in italics in syntax diagrams in an alternate path from the preceding term along with other optional values or a default value.
DefaultValue	A default value is the value that is specified by default for a term. A default value is shown in italics in syntax diagrams in an alternate path above other optional values.

Configuring an IMS Client

Summary Table of Valid Servers and Protocols

Table 17. Valid Servers and Protocols for VisualAge Generator Clients on the IMS Platform

Server Platforms	Protocols for IMS Client Platform
AIX	N/A
CICS for AIX	N/A
HP-UX	N/A
IMS	N/A
CICS for MVS/ESA	N/A
OS/2	N/A
CICS for OS/2	N/A
OS/400	N/A
Solaris	N/A
CICS for Solaris	N/A
VM/ESA	N/A
CICS for VSE/ESA	N/A
Windows NT (C++)	N/A
Windows NT (Java)	N/A
CICS for Windows NT	N/A
Notes: N/A = Not available	

Note: The IMS platform does not support IMS clients.

Configuring an IMS Server

Summary Table of Valid Clients and Protocols

Table 18. Valid Clients and Protocols for VisualAge Generator Servers on the IMS Platform

Server Platforms	**Client Platforms**		
	OS/2 (GUI, C++, ITF)	**Windows 95 and Windows 98 (GUI)**	**Windows NT (GUI, C++, ITF)**
IMS	APPC/IMS	APPC/IMS	APPC/IMS

APPC/IMS Protocol

List of Valid Clients

- OS/2 (GUI, C++, ITF)
- Windows 95 and Windows 98 (GUI)
- Windows NT (GUI, C++, ITF)

Identifying the Server Location

Use CPIC side information on the client to specify the following information so that the client can allocate the APPC session:

- Partner LU alias
- Transaction program name
- Mode name

The partner LU alias must be the APPC LU name of the IMS Transaction Manager that you want to run the server program. The server program is defined to MVS/APPC.

The following is an example of the command needed to add a transaction program name to APPC/MVS. In this example, tpname must match the transaction program name in the CPIC side profile on the client.

```
TPADD TPSCHED_EXIT(DFSTPPE0)
TPNAME(tpname)
SYSTEM
ACTIVE(YES)
TPSCHED_DELIMITER(##)
TRANCODE=trancode
#
```

Specify the side information identifier in the `location` linkage table attribute for the server program, or set the location dynamically in the client program at run time using EZELOC.

Linkage Table Attributes for Generating IMS Server Programs

No linkage table entry is required when generating the server program. If an entry specifying:

```
►►──linktype=───REMOTE─────remotecomtype=───APPCIMS──────────────────────────────►◄
```

is included, the program is generated like a program with:

```
►►──linktype=───DYNAMIC──parmform=───OSLINK──────────────────────────────────────►◄
```

Server Program Set Up and Operation

The server program is generated and prepared as a called batch program on the server system.

A server transaction must be defined to the IMS system. The transaction is defined as an asynchronous Message Processing Program (MPP). The following IMS system definition parameters are required for the definition:

```
APPLCTN PGMTYPE=TP,PSB=ims-psb-name
TRANSACT CODE=trancode,MODE=SNGL,EDIT=ULC
```

Note: Data will be folded to uppercase characters if the statement EDIT=ULC is omitted from the transaction definition.

A catcher program, ELAISVN, provided with VisualAge Generator Server for MVS, VSE, and VM is the first program called for the transaction. The catcher program must be relinked with an alias that is the same as the IMS PSB name associated with the transaction. An example of the JCL used to relink ELAISVN follows.

```
//L    EXEC ELARLINK
//L.SYSLMOD DD DISP=SHR,DSN=load-library-name
//L.SYSIN DD *
 INCLUDE SELALMD(ELAISVN)
 ENTRY ELAISVN
 ALIAS ims-psb-name
 NAME load-module-name(R)
/*
```

Up to 64 aliases can be associated with one load module. Include ALIAS cards for all aliases whenever you relink to add additional aliases.

The client program communicates with APPC so that the call parameters are written as a message to the IMS message queue associated with the transaction. The catcher program reads the messages from the queue and calls the server program. When the server program returns, the catcher program writes the modified parameters back to the output message queue from which APPC returns them to the client program.

Chapter 10. CICS for MVS/ESA Platform

How to use this chapter

This chapter contains platform and protocol information necessary to configure a client program or a server program. The chapter is divided into configuration sections: client programs and server programs. Each configuration section is organized by protocol. Each protocol section contains information organized by target platforms.

Use the following method to locate the information you need:

1. Verify that this chapter is appropriate to your runtime environment.

   Ensure that the runtime environment of your program matches the platform on which this chapter is based. Turn to the appropriate chapter if the runtime environment of your program does not match the platform named in the title of this chapter.
2. Determine which type of program you want to configure: client program or server program.
3. Determine which configuration section in the chapter is appropriate for your program.
4. Find the intended target platform of the program you are configuring in the Summary Table of Valid Clients and Protocols or Summary Table of Valid Servers and Protocols.

   If you are configuring a client program, find the target platform of the intended server program in the column of valid platforms.

   If you are configuring a server program, find the target platform of the intended client program in the row of valid platforms.
5. Chose a protocol based on the information in the table.

   If you are configuring a client program, find the cell in the table where the row of the intended target platform intersects the column of the client platform. Select a protocol from the list of available protocols for this client/server combination.

   If you are configuring a client program, find the cell in the table where the column of the intended target platform intersects the row of the server platform. Select a protocol from the list of available protocols for this client/server combination.
6. Determine the name of the appropriate protocol section within the configuration section.

Turn to the part of the configuration section indicated by the protocol name. Here you will find information common to all target platforms valid for this protocol.

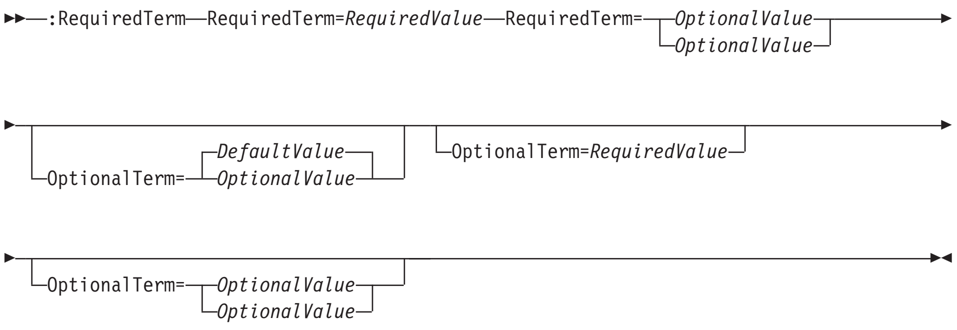
7. Determine the name of the appropriate target platform section within the protocol section.

   You will find platform specific information for the chosen protocol in the section named for your intended target platform.

Example

```
Chapter: CICS for MVS/ESA Platform
Type of program to configure: Client program
Configuration section: Configuring a CICS for MVS/ESA client
Intended target platform: CICS for AIX (i.e. a CICS for AIX server program)
Chosen protocol: CICS DPL
Protocol section: CICS DPL Protocol
Target platform section: Configuring a CICS for MVS/ESA Client for a
CICS for AIX Server
```

You will encounter syntax diagrams in this chapter. The diagrams are used to show programming syntax as follows:



RequiredTerm	A required term is a term that must be specified. Required terms are depicted in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required terms.
RequiredValue	A required value is a value that must be specified for a term. A required value is shown in italics in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required values.
OptionalTerm	An optional term is a term that can be

	specified. An optional term is depicted in syntax diagrams in an alternate path from the preceding term.
OptionalValue	An optional value is a value that can be specified for a term. An optional value is shown in italics in syntax diagrams in an alternate path from the preceding term along with other optional values or a default value.
DefaultValue	A default value is the value that is specified by default for a term. A default value is shown in italics in syntax diagrams in an alternate path above other optional values.

Configuring a CICS for MVS/ESA Client

Summary Table of Valid Servers and Protocols

Table 19. Valid Servers and Protocols for VisualAge Generator Clients on the CICS for MVS/ESA Platform

Server Platforms	Protocols for CICS for MVS/ESA Client Platform
CICS for AIX	CICS DPL
CICS for MVS/ESA	CICS DPL
CICS for OS/2	CICS DPL
CICS for Solaris	CICS DPL
CICS for VSE/ESA	CICS DPL
CICS for Windows NT	CICS DPL

CICS DPL Protocol

User Authentication

The user associated with the CICS client transaction must be authorized to run the client transaction and the server transaction. The user ID and password are identified by the CICS transaction manager on the client system.

Setting Up Communication Links

The communication link between the client and server systems must be defined to the CICS systems to allow a distributed program link from one system to the other. Refer to the CICS intercommunication manual for your CICS systems for more information.

Controlling the Unit of Work

Use the `luwcontrol` linkage table attribute to indicate whether the server updates are automatically committed on return or the client controls the unit of work. If the client is controlling the unit of work, subsequent server calls must go to the same system and transaction under client-controlled unit of work until a commit or roll back is requested. The server cannot issue EZECOMIT or EZEROLLB calls if client-controlled unit of work is specified.

Data Format Conversion

Code format conversion is performed on the server call as specified in the `contable` linkage attribute table and in EZECONVT. Code is converted based on the structure of the arguments that are specified on the CALL statement in the client program.

Error Handling

The standard error handling procedures described below are supported with CICS DPL. If the DPL is not successful for any reason, including unavailability of the communication link, error information is returned with CICS in the CICS EIB (EXEC interface block).

If the REPLY option is not specified on the CALL statement, the calling program ends with error messages. If REPLY is specified, no messages are written or logged, and EZERT8 is set to one of the following values:

Value	Meaning
00000000	Successful call and return
00000204	Program name not valid
00000207	System identifier not valid
00000208	Link out of service
ffrrrrrr	Other CICS error where ff is the hexadecimal representation of EIBFN byte 0, and rrrrrr is the hexadecimal representation of EIBRCODE bytes 0–2.

Configuring a CICS for MVS/ESA Client for a CICS for OS/2 Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►—:calllink—applname=program name—linktype=—REMOTE—┬──────────────────────┬──────────►
                                                     └─parmform=—COMMDATA───┘
```

```
►─┬───────────────────────────────────────┬──────────────────────────────────────────────────►
  │              ┌─GENERATION─┐           │
  └─remotebind=──┴────────────┴─RUNTIME───┘

►─┬───────────────────────────────────────┬─┬─────────────────────────────┬────────────────────►
  └─contable=──┬─conversion table name─┬──┘ └─location=──┬─EZELOC──────┬──┘
               ├─'*' ──────────────────┤                 └─system name─┘
               ├─EZECONVT──────────────┤
               ├─NONE──────────────────┤
               └─BINARY────────────────┘

►─┬─────────────────────────────┬─┬──────────────────────────────┬─────────────────────────────►
  │              ┌─CLIENT─┐     │ └─remoteapptype=──┬─VG────┬────┘
  └─luwcontrol=──┴─SERVER─┴─────┘                   ├─NONVG─┤
                                                    └─ITF───┘

►─┬────────────────────────────────────┬──────────────────────────────────────────────────────►◄
  └─serverid=──server identifier───────┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CSO7 contable=ELACNENU LUWCONTROL=CLIENT
```

Configuring a CICS for MVS/ESA Client for a CICS for Windows NT Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►──:calllink──applname=program name──linktype=─REMOTE──┬──────────────────────┬────────────►
                                                        └─parmform=─COMMDATA───┘

►─┬───────────────────────────────────────┬──────────────────────────────────────────────────►
  │              ┌─GENERATION─┐           │
  └─remotebind=──┴────────────┴─RUNTIME───┘
```

```
►─┬──────────────────────────────────────────┬──┬─────────────────────────────────┬──────────►
  └─contable=──┬─conversion table name─┬─────┘  └─location=──┬─EZELOC──────┬─────┘
               ├─'*' ──────────────────┤                    └─system name─┘
               ├─EZECONVT──────────────┤
               ├─NONE──────────────────┤
               └─BINARY────────────────┘

►─┬────────────────────────────────────┬─┬─────────────────────────────────┬──────────────────►
  │                  ┌─CLIENT─┐        │ └─remoteapptype=──┬─VG────┬───────┘
  └─luwcontrol=──────┴─SERVER─┴────────┘                   ├─NONVG─┤
                                                           └─ITF───┘

►─┬──────────────────────────────────────────┬─────────────────────────────────────────────►◄
  └─serverid=──server identifier─────────────┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CPMI contable=ELACNENU LUWCONTROL=CLIENT
```

Configuring a CICS for MVS/ESA Client for a CICS for AIX Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►──:calllink──applname=program name──linktype=─REMOTE──┬─────────────────────┬──────────►
                                                        └─parmform=─COMMDATA──┘

►─┬────────────────────────────────────────────────┬─────────────────────────────────────►
  │             ┌─GENERATION──┐                    │
  └─remotebind=─┴─────────────┴──RUNTIME───────────┘

►─┬──────────────────────────────────────────┬──┬─────────────────────────────────┬──────────►
  └─contable=──┬─conversion table name─┬─────┘  └─location=──┬─EZELOC──────┬─────┘
               ├─'*' ──────────────────┤                    └─system name─┘
               ├─EZECONVT──────────────┤
               ├─NONE──────────────────┤
               └─BINARY────────────────┘
```

```
►─┬──────────────────────────────┬─┬──────────────────────────────┬──────────────►
  │               ┌─CLIENT─┐     │ └─remoteapptype=──┬─VG────┬────┘
  └─luwcontrol=───┴─SERVER─┴─────┘                   ├─NONVG─┤
                                                     └─ITF───┘

►─┬─────────────────────────────────────┬──────────────────────────────────────►◄
  └─serverid=──server identifier────────┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CS07 contable=ELACNENU LUWCONTROL=CLIENT
```

Configuring a CICS for MVS/ESA Client for a CICS for MVS/ESA Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►──:calllink──applname=program name──linktype=─REMOTE─┬──────────────────────┬──►
                                                       └─parmform=─COMMDATA───┘

►─┬───────────────────────────────────────────┬──────────────────────────────────►
  │               ┌─GENERATION─┐              │
  └─remotebind=───┴────────────┴───RUNTIME────┘

►─┬──────────────────────────────────────┬─┬─────────────────────────────────┬──►
  └─contable=──┬─conversion table name───┘ └─location=──┬─EZELOC──────────┬──┘
               ├─'*'─────────────────────┤              └─system name─────┘
               ├─EZECONVT────────────────┤
               ├─NONE────────────────────┤
               └─BINARY──────────────────┘

►─┬──────────────────────────────┬─┬──────────────────────────────┬──────────────►
  │               ┌─CLIENT─┐     │ └─remoteapptype=──┬─VG────┬────┘
  └─luwcontrol=───┴─SERVER─┴─────┘                   ├─NONVG─┤
                                                     └─ITF───┘
```

```
►─┬───────────────────────────────┬──────────────────────────────────────────►◄
  └─serverid=──server identifier──┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CS07 LUWCONTROL=CLIENT
```

Configuring a CICS for MVS/ESA Client for a CICS for VSE/ESA Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►──:calllink──applname=program name──linktype=─REMOTE─┬──────────────────────┬──────►
                                                       └─parmform=─COMMDATA───┘

►─┬──────────────────────────────────────┬────────────────────────────────────────►
  │              ┌─GENERATION─┐          │
  └─remotebind=──┴────────────┴─RUNTIME──┘

►─┬─────────────────────────────────────────┬─┬───────────────────────────┬───────►
  └─contable=──┬─conversion table name──┬───┘ └─location=──┬─EZELOC──────┬─┘
               ├─'*'────────────────────┤                 └─system name─┘
               ├─EZECONVT───────────────┤
               ├─NONE───────────────────┤
               └─BINARY─────────────────┘

►─┬───────────────────────────┬─┬─────────────────────────────┬───────────────────►
  │              ┌─CLIENT─┐   │ └─remoteapptype=──┬─VG────┬───┘
  └─luwcontrol=──┴─SERVER─┴───┘                   ├─NONVG─┤
                                                  └─ITF───┘

►─┬───────────────────────────────┬──────────────────────────────────────────►◄
  └─serverid=──server identifier──┘
```

The following attributes are ignored:
- externalname
- library

- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CS07 LUWCONTROL=CLIENT
```

Configuring a CICS for MVS/ESA Client for a CICS for Solaris Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►—:calllink—applname=program name—linktype=—REMOTE—┬────────────────────┬──────►
                                                    └parmform=—COMMDATA──┘


►─┬──────────────────────────────────────┬────────────────────────────────────────►
  │              ┌GENERATION─┐           │
  └remotebind=───┴───────────┴──RUNTIME──┘


►─┬────────────────────────────────────┬─┬───────────────────────────┬────────────►
  └contable=──┬conversion table name──┤ └location=──┬EZELOC──────┬──┘
              ├'*' ───────────────────┤             └system name─┘
              ├EZECONVT───────────────┤
              ├NONE───────────────────┤
              └BINARY─────────────────┘


►─┬─────────────────────────┬─┬────────────────────────────────┬──────────────────►
  │              ┌CLIENT─┐  │ └remoteapptype=──┬VG─────┬───────┘
  └luwcontrol=───┴SERVER─┴──┘                  ├NONVG──┤
                                               └ITF────┘


►─┬─────────────────────────────────┬─────────────────────────────────────────────►◄
  └serverid=───server identifier────┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CS07 contable=ELACNENU LUWCONTROL=CLIENT
```

Configuring a CICS for MVS/ESA Server

Summary Table of Valid Clients and Protocols

Table 20. Valid Clients and Protocols for VisualAge Generator Servers on the CICS for MVS/ESA Platform

Server Platforms	**Client Platforms**					
	OS/2 (GUI, C++, ITF)	**Windows 95 and Windows 98 (GUI)**	**Windows NT (GUI, C++, ITF)**	**AIX (C++)**	**CICS**	**Solaris (C++)**
CICS for MVS/ESA	CICS Client, LU2	CICS Client	CICS Client	CICS Client	CICS DPL	CICS Client

Notes:

Client controlled unit of work is available for CICS Client, LU2, and CA/400.
N/A = Not available
APPC = Advanced Program-to-Program Communication.
All server environments support server controlled unit of work.
CICS as a Client Platform refers to:
- CICS for AIX
- CICS for MVS/ESA
- CICS for OS/2
- CICS for Windows NT
- CICS for VSE/ESA
- CICS for Solaris

CICS Client Protocol

List of Valid Clients

- OS/2 (GUI, C++, ITF)
- Windows 95 and Windows 98 (GUI)
- Windows NT (GUI, C++, ITF)
- AIX (C++)
- Solaris (C++)

Identifying the Server Location

Specify the server location (system identifier) in the LOCATION linkage table attribute. You can also set the location dynamically in the client program at run time using EZELOC. If the server location is not specified, the default is the first entry in the CICS Client initialization file.

Linkage Table Attributes for Generating CICS Server Programs

Specify the following calllink attributes when you generate CICS server programs:

```
►►─:calllink─applname=program name─linktype=─REMOTE─┬──────────────────────┬──────►◄
                                                    └─parmform=─COMMDATA───┘
```

The following attributes are ignored:
- contable
- externalname
- library
- location
- luwcontrol
- remoteapptype
- remotebind
- remotecomtype
- serverid

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA
```

Considerations for Defining the Server Program

If the remote program performs printing, it must move the print destination into EZEDESTP.

Customizing CICS for MVS/ESA for Called VAGen Server Programs

Calling a CICS for MVS/ESA server program that is specified remote in the linkage tables requires that the called server program be defined to CICS/ESA. Either the batch program DFHCSDUP utility or the resource definition online (RDO) CEDA DEFINE PROGRAM command can be used to define the VAGen server program to MVS CICS.

Example

```
CEDA DEFINE PROGRAM(applname)
GROUP(cscgroup) LANGUAGE(COBOL)
```

The CICS supplied mirror transaction DFHMIRS, normally invoked by the CPMI transaction, uses the data passed to it by the client to:
1. Determine which server program should be given control
2. Build the COMMAREA
3. Link to the defined VAGen server program via CICS LINK

CPMI is the CICS supplied default transaction code to invoke the CICS mirror program DFHMIRS. CPMI must be defined with specified values if used for calling VAGen server applications.

It is recommended that you copy the CICS supplied CPMI definitions to a new group to make the changes, then edit the CPMI transaction to:
1. Make the twasize 1024

2. Make the profile DFHCICSA (CICS default would be DFHCICST (T for terminal))
3. Ensure that the program pointed to remains DFHMIRS

As CPMI is the default transaction name associated with a VAG CICS server program, the linkage table used does not need to specify serverid=CPMI although it is advisable to use caution when relying on defaults.

If CPMI is not used, then define a transaction code of your choice using the above entries.

Example

```
PROGRAM(DFHMIRS) TWASIZE(1024)
PROFILE(DFHCICSA)
```

Relating the CICS Application to a DB2 Plan: A DB2 authorization ID and a DB2 plan are required if a CICS application accesses DB2. The DSNRCT macro in the CICS resource control table (RCT) relates a CICS transaction with the DB2 plan name used by the application and a DB2 authorization ID.

The CPMI transaction can be used or additional transactions invoking the mirror program can be specified. Copy the CPMI definitions to ensure these additional transactions have the correct specifications.

For VAGen, the transaction code invoking the mirror program and identified in the RCT corresponds to the `serverid` field in the linkage table entry for the server application.

The following example is a DSNCRCT macro which relates the transaction code CPMI with the DB2 plan DBSERVE. The signed-on user ID is the authorization ID.

Example

```
DSNCRCT TYPE=ENTRY,
PLAN=DBSERVE,
AUTH=(USERID,*,*),
THRDM=1,
THRDA=1,
TXID=(CPMI,cpmi)
```

In the case where an additional or alternate transaction code is defined such as:

```
DEFINE TRANSACTION(ABCD) GROUP(CICSGROUP)
PROGRAM(DFHMIRS) TWASIZE(1024)
PROFILE(DFHCICSA)
```

then the RCT entry would be:

```
DSNCRCT TYPE=ENTRY,
PLAN=DBSERVE,
AUTH=(USERID,*,*),
THRDM=1,
THRDA=1,
TXID=(ABCD,abcd)
```

The transaction invoking DFHMRIS and identified in the RCT is used for the `serverid` field in the linkage table entry for the server application.

Providing Security for CICS Application Programs: You need to consider the issue of security with application programs. A client program is allowed to invoke the mirror program if any of the following is true:

- Resource access control facility (RACF) security is not used
- The transaction for the mirror program (either CPMI or the alternate transaction ID) does not have a RACF profile
- The transaction for the mirror program (either CPMI or the alternate transaction ID) has a RACF profile with universal access of read

If RACF security is used to control access to application programs, certain CICS and RACF administration activities are required:

- Set the XTRAN parameter in the system initialization table (SIT) to YES
- Create a profile in RACF class TCICSTRN for each unique mirror transaction
- Provide read access to the RACF profile for the mirror transaction for the following:
  - The CICS system default user ID(DFLTUSER)
  - Any user ID used as "Securityname" for a LU 6.2 connection
  - Any user ID forwarded from a client program

LU 6.2 Security: For LU 6.2 connections, if RACF is active the connections must be defined with "ATtachsec" set to "Verify" in order to have the user ID and password verified at transaction attachment. The authentication exit for the client obtains a user ID and password and forwards them to the CICS for MVS/ESA system for verification. The mirror transaction is initiated and the application program is invoked if the user ID and password are verified and the user ID had read access to the profile for the transaction. Otherwise, an error is returned to the client.

Specifying Parameter Format

Parameter format (the `parmform` attribute on the calllink tag) can be specified as COMMDATA (data values passed in the COMMAREA) or COMMPTR (pointers passed in the COMMAREA).

Using COMMPTR Format: Use COMMPTR for calling COBOL subroutines that were developed to use the non-VAGen parameter passing convention that uses pointers in the CICS COMMAREA. When the pointer format is used, the generated client program calls an intermediate catcher program (ELACSV, provided with VisualAge Generator host services) via the ECI instead of calling the server program directly. The catcher program receives the parameter data from the client, builds a pointer list in the COMMAREA that points to the individual parameters, and links to the server program. A program processing table (PPT) must be defined for the ELACSV program on the CICS server system.

COMMPTR format is only supported for MVS CICS, CICS for VSE/ESA, and CICS OS/2 servers.

Using COMMDATA Format: Use COMMDATA for calling generated programs and for calling COBOL subroutines where possible. COMMDATA provides better performance in calling remote programs. With COMMDATA, the server program is called directly without going through the catcher program.

CICS DPL Protocol

List of Valid Clients

- CICS for AIX
- CICS for MVS/ESA
- CICS for OS/2
- CICS for Windows NT
- CICS for VSE/ESA
- CICS for Solaris

Identifying the Server Location

You can specify the server location (CICS system identifier) on the `location` linkage table attribute, or dynamically set the location in the client program at runtime using EZELOC. If the location is not specified, the default location is the system identifier associated with the server program in the CICS program definition on the client system.

Linkage Table Attributes for Generating CICS Server Programs

Specify the following calllink attributes when you generate CICS server programs:

```
►►—:calllink—applname=program name—linktype=—REMOTE——┬———————————————————┬————————►◄
                                                      └parmform=—COMMDATA—┘
```

The following attributes are ignored:

- contable
- externalname
- library

- location
- luwcontrol
- remoteapptype
- remotebind
- remotecomtype
- serverid

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA
```

Considerations for Defining the Server Program

If the remote program performs printing, it must move the print destination into EZEDESTP.

Server Program Set Up

The server program is generated and prepared as is any other CICS program on the server system. In addition, there must be a CICS program definition for the server program on the CICS client system containing the following information:
- Resident option = remote
- Remote system identifier
- Remote program name (optional)
- Remote transaction identifier (optional)

The server transaction must be defined to CICS on the server system. The default transaction name associated with a server program is the CICS-supplied mirror transaction, CPMI.

LU2 Protocol

List of Valid Clients

- OS/2 (GUI, C++, ITF)

Linkage Table Attributes for Generating Server Programs

New section...

Example

```
:calllink applname=ELACVP5 parmform=COMMDATA linktype=REMOTE remotecomtype=LU2
   location=CICSTST
```

Customizing an MVS CICS Communication Server for LU2

The following customizations must be completed before any service calls are requested by a GUI or native C++ program.

Defining Programs to MVS CICS for LU2: The VisualAge Generator program specified as REMOTE in the linkage table must be defined to MVS CICS before it can be accessed by a GUI or native C++ client program. The

VisualAge Generator client/server communication program, ELACLU2, shipped with VisualAge Generator MVS Host Services PTF UQ09122 or VisualAge Generator Server for MVS, VSE, and VM Version 1.2 is defined to MVS CICS during installation. It should be defined with a transaction ID of your choice and with a transaction work area (TWA) size of 1024. On MVS CICS, the ELACLU2 program uses the data passed to it by the client to determine which server program should be given control, builds the COMMAREA, and links to the VisualAge Generator server program. Because the ELACLU2 program links to the generated server program, the VisualAge Generator server program name must be defined to MVS CICS. Either the batch DFHCSDUP utility or the resource definition online (RDO) CEDA DEFINE PROGRAM command can be used to define the VisualAge Generator server program to MVS CICS.

The following example shows the DFHCSDUP command for defining a VisualAge Generator remote server program to MVS CICS:

```
DEFINE PROGRAM(applname) GROUP(cscgroup) LANGUAGE(COBOL)
```

Defining Programs to MVS CICS for LU2: A DB2 authorization ID and a DB2 plan are required if a VisualAge Generator remote server program accesses DB2 tables. The DSNCRCT macro in the CICS resource control table (RCT) relates a CICS transaction code with the DB2 plan name used by the program and a DB2 authorization ID. The transaction code identified in the RCT is used for the Server ID field in the communication client's linkage table entry for the server program. You need to define an alternate transaction code for the VisualAge Generator client/server communication program ELACLU2 to create a unique transaction code that matches the transaction name in the linkage table.

The following example shows how to define an alternate transaction for the communication program ELACLU2. The alternate transaction name must have a transaction work area (TWA) size of 1024 bytes.

```
DEFINE TRANSACTION(DBSV) GROUP(cscgroup) PROGRAM(ELACLU2)
                   TWASIZE(1024)
```

The following example shows a sample DSNCRCT macro which relates the alternate code (DBSV in this example) with the DB2 plan DBSERVE. The signed-on user ID is the authorization ID.

```
DSNCRCT TYPE=ENTRY,                         C
      PLAN=DBSERVE,                         C
      AUTH=(USERID,*,*),                    C
      THRDM=1,                              C
      THRDA=1,                              C
      TXID=(DBSV,dbsv)
```

Chapter 11. Solaris Platform

How to use this chapter

This chapter contains platform and protocol information necessary to configure a client program or a server program. The chapter is divided into configuration sections: client programs and server programs. Each configuration section is organized by protocol. Each protocol section contains information organized by target platforms.

Use the following method to locate the information you need:

1. Verify that this chapter is appropriate to your runtime environment.

   Ensure that the runtime environment of your program matches the platform on which this chapter is based. Turn to the appropriate chapter if the runtime environment of your program does not match the platform named in the title of this chapter.
2. Determine which type of program you want to configure: client program or server program.
3. Determine which configuration section in the chapter is appropriate for your program.
4. Find the intended target platform of the program you are configuring in the Summary Table of Valid Clients and Protocols or Summary Table of Valid Servers and Protocols.

   If you are configuring a client program, find the target platform of the intended server program in the column of valid platforms.

   If you are configuring a server program, find the target platform of the intended client program in the row of valid platforms.
5. Chose a protocol based on the information in the table.

   If you are configuring a client program, find the cell in the table where the row of the intended target platform intersects the column of the client platform. Select a protocol from the list of available protocols for this client/server combination.

   If you are configuring a client program, find the cell in the table where the column of the intended target platform intersects the row of the server platform. Select a protocol from the list of available protocols for this client/server combination.
6. Determine the name of the appropriate protocol section within the configuration section.

Turn to the part of the configuration section indicated by the protocol name. Here you will find information common to all target platforms valid for this protocol.

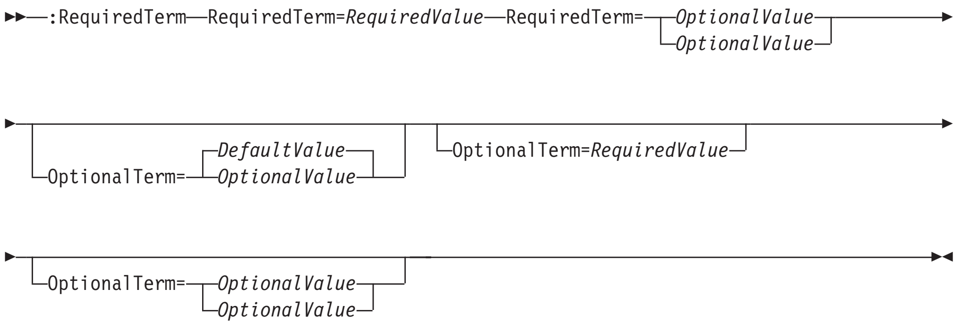
7. Determine the name of the appropriate target platform section within the protocol section.

   You will find platform specific information for the chosen protocol in the section named for your intended target platform.

Example

```
Chapter: Solaris Platform
Type of program to configure: Server program
Configuration section: Configuring a Solaris server
Intended target platform: OS/2 (i.e. an OS/2 client program)
Chosen protocol: TCP/IP
Protocol section: TCP/IP Protocol
```

You will encounter syntax diagrams in this chapter. The diagrams are used to show programming syntax as follows:



RequiredTerm	A required term is a term that must be specified. Required terms are depicted in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required terms.
RequiredValue	A required value is a value that must be specified for a term. A required value is shown in italics in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required values.
OptionalTerm	An optional term is a term that can be

	specified. An optional term is depicted in syntax diagrams in an alternate path from the preceding term.
OptionalValue	An optional value is a value that can be specified for a term. An optional value is shown in italics in syntax diagrams in an alternate path from the preceding term along with other optional values or a default value.
DefaultValue	A default value is the value that is specified by default for a term. A default value is shown in italics in syntax diagrams in an alternate path above other optional values.

Configuring a Solaris Client

Summary Table of Valid Servers and Protocols

Table 21. Valid Servers and Protocols for VisualAge Generator Clients on the Solaris Platform

Server Platforms	Protocols for Solaris Client Platform
AIX	TCP/IP
CICS for AIX	CICS Client
HP-UX	TCP/IP
CICS for MVS/ESA	CICS Client
OS/2	TCP/IP
CICS for OS/2	CICS Client
Solaris	TCP/IP, IPC, DIRECT
CICS for Solaris	CICS Client
VM/ESA	TCP/IP
CICS for VSE/ESA	CICS Client
Windows NT	TCP/IP
Windows NT (Java)	N/A
CICS for Windows NT	CICS Client

CICS Client Protocol

User Authentication

The user authentication exit (see "User Authentication" on page 21) provides the user ID and the password specified on the ECI call. The program user must be authorized to run the transaction associated with the server call.

The user exit can return NULLs for the userid and password. The default exit returns the contents of the environment variables CSOUID and CSOPWD as the userid and password. If nulls are specified on the ECI call, the CICS Client determines the user ID and password in a system-dependent fashion. Refer to the CICS Client documentation for your environment for further information.

Setting Up Communication Links

The communication link between the client and server systems must be defined to the CICS server system and client products to allow an ECI call to flow from a CICS Client product to server systems. CICS Client, the communications software, is installed and configured on the client. Refer to CICS CLIENT documentation. The CICS server environment must have a "listener" defined and requires other entries if remote programs are called. Refer to CICS documentation for more information. Refer to the CICS intercommunication documentation for your CICS systems and client products for additional information.

Identifying the CICS Transaction for the Server

The CICS transaction name associated with the server program is specified in the `serverid` linkage table attribute. If not specified, the default transaction is the CICS-supplied mirror transaction, CPMI.

Controlling the Unit of Work

Extended Units of Work: Multiple synchronous calls to CICS servers can be issued from the same client. You can use the extended unit of work feature of ECI to include several calls to the same system within the same unit of work by specifying `luwcontrol=CLIENT` in the linkage table for the server programs. For CICS servers, the default value for LUWCONTROL is client unit of work. The extended unit of work ends when the client program calls EZECOMIT or EZEROLLB, which results in an ECI call to commit or roll back any extended transactions that are currently active.

A separate ECI extended unit of work (CICS transaction) is started for each unique `serverid/location` pair. Servers on the same system running under the same SERVERID (transaction name) are part of the same CICS extended transaction. A client EZECOMIT or ROLLBACK call ends all the extended transactions currently in effect for the client.

The server cannot issue EZECOMIT or EZEROLLB calls if the client unit of work was in effect for the server call.

Server Unit of Work: You can specify `luwcontrol=SERVER` in the linkage table. With this specification, each server call is a separate unit of work. The server program can issue commit or rollback requests as well.

Data Format Conversion

Code format conversion is performed on the server call as specified on the `contable` linkage table attribute and EZECONVT special function word. Code is converted based on the structure of the arguments specified on the CALL statement in the client program.

Error Handling

The standard error handling procedures described in “Communication Error Handling” on page 22 are supported with CICS ECI. EZERT8 is set to the decimal value of the client access service reason code. The reason code is the same as the number portion of error message CSOnnnna as listed in the *VisualAge Generator Messages and Problem Determination Guide* document.

Specifying Parameter Format

Parameter format (the `parmform` attribute on the calllink tag) can be specified as COMMDATA (data values passed in the COMMAREA) or COMMPTR (pointers passed in the COMMAREA).

Using COMMPTR Format: Use COMMPTR for calling COBOL subroutines that were developed to use the non-VAGen parameter passing convention that uses pointers in the CICS COMMAREA. When the pointer format is used, the generated client program calls an intermediate catcher program (ELACSV, provided with VisualAge Generator host services) via the ECI instead of calling the server program directly. The catcher program receives the parameter data from the client, builds a pointer list in the COMMAREA that points to the individual parameters, and links to the server program. A program processing table (PPT) must be defined for the ELACSV program on the CICS server system.

COMMPTR format is only supported for MVS CICS, CICS for VSE/ESA, and CICS OS/2 servers.

Using COMMDATA Format: Use COMMDATA for calling generated programs and for calling COBOL subroutines where possible. COMMDATA provides better performance in calling remote programs. With COMMDATA, the server program is called directly without going through the catcher program.

Configuring a Solaris Client for a CICS for OS/2 Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►—:calllink—applname=program name—linktype=–REMOTE—remotecomtype=–CICSCLIENT——————►
```

```
►──┬─────────────────────────────────────────┬───┬────────────────────────────────┬──────►
   └─contable=──┬─conversion table name─┬───┘   └─location=──┬─EZELOC──────┬───┘
                ├─'*' ──────────────────┤                     └─system name─┘
                ├─EZECONVT──────────────┤
                ├─NONE──────────────────┤
                └─BINARY────────────────┘

►──┬─────────────────────────────┬──┬──────────────────────────────┬──────────────►
   │             ┌─CLIENT─┐      │  └─parmform=──┬─OSLINK────┬────┘
   └─luwcontrol=─┴─SERVER─┴──────┘               ├─COMMPTR───┤
                                                 ├─COMMDATA──┤
                                                 └─CICSOSLINK┘

►──┬────────────────────────────┬──┬───────────────────────────────────┬──────────►
   └─remoteapptype=──┬─VG────┬──┘  │               ┌─GENERATION─┐      │
                     ├─NONVG─┤     └─remotebind=───┴─RUNTIME────┴──────┘
                     └─ITF───┘

►──┬──────────────────────────────────┬───────────────────────────────────────────►◄
   └─serverid=──server identifier─────┘
```

The following attributes are ignored:
- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST contable=BINARY
```

Configuring a Solaris Client for a CICS for Windows NT Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─CICSCLIENT──────►

►──┬─────────────────────────────────────────┬───┬────────────────────────────────┬──────►
   └─contable=──┬─conversion table name─┬───┘   └─location=──┬─EZELOC──────┬───┘
                ├─'*' ──────────────────┤                     └─system name─┘
                ├─EZECONVT──────────────┤
                ├─NONE──────────────────┤
                └─BINARY────────────────┘
```

```
►─┬─────────────────────────────┬─┬─────────────────────────────┬──────────────►
  │               ┌─CLIENT─┐    │ └─parmform=─┬─OSLINK─────┬─┘
  └─luwcontrol=───┴─SERVER─┴────┘             ├─COMMDATA───┤
                                              └─CICSOSLINK─┘

►─┬───────────────────────────────┬─┬───────────────────────────────┬──────────►
  └─remoteapptype=─┬─VG────────┬──┘ │               ┌─GENERATION─┐  │
                   ├─NONVG─────┤    └─remotebind=───┴─RUNTIME────┴──┘
                   └─ITF───────┘

►─┬──────────────────────────────────┬────────────────────────────────────────►◄
  └─serverid=───server identifier────┘
```

The following attributes are ignored:
- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST contable=BINARY
```

Configuring a Solaris Client for a CICS for AIX Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─CICSCLIENT──────►

►─┬─────────────────────────────────────────┬─┬───────────────────────────────┬──────►
  └─contable=─┬─conversion table name───────┴─┘ └─location=─┬─EZELOC──────┬────┘
              ├─'*' ────────────────────────┤               └─system name─┘
              ├─EZECONVT────────────────────┤
              ├─NONE────────────────────────┤
              └─BINARY──────────────────────┘

►─┬─────────────────────────────┬─┬─────────────────────────────┬──────────────►
  │               ┌─CLIENT─┐    │ └─parmform=─┬─OSLINK─────┬─┘
  └─luwcontrol=───┴─SERVER─┴────┘             ├─COMMDATA───┤
                                              └─CICSOSLINK─┘
```

```
►──┬──────────────────────────────┬──┬────────────────────────────────────┬──────────────────────►
   └─remoteapptype=──┬─VG────┬────┘  │                ┌─GENERATION──┐     │
                     ├─NONVG─┤       └─remotebind=────┴─RUNTIME─────┴─────┘
                     └─ITF───┘

►──┬──────────────────────────────────┬───────────────────────────────────────────────────────►◄
   └─serverid=──server identifier─────┘
```

The following attributes are ignored:
- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST contable=BINARY
```

Configuring a Solaris Client for a CICS for MVS/ESA Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─CICSCLIENT──────────►

►──┬─────────────────────────────────────────┬──┬──────────────────────────────────┬──────────►
   └─contable=──┬─conversion table name─┬────┘  └─location=──┬─EZELOC──────┬──────┘
                ├─'*'───────────────────┤                    └─system name─┘
                ├─EZECONVT──────────────┤
                ├─NONE──────────────────┤
                └─BINARY────────────────┘

►──┬───────────────────────────────────┬──┬────────────────────────────────────┬─────────────►
   │                  ┌─CLIENT─┐       │  └─parmform=──┬─OSLINK────┬──────────┘
   └─luwcontrol=──────┴─SERVER─┴───────┘               ├─COMMPTR───┤
                                                       ├─COMMDATA──┤
                                                       └─CICSOSLINK┘

►──┬──────────────────────────────┬──┬────────────────────────────────────┬──────────────────────►
   └─remoteapptype=──┬─VG────┬────┘  │                ┌─GENERATION──┐     │
                     ├─NONVG─┤       └─remotebind=────┴─RUNTIME─────┴─────┘
                     └─ITF───┘

►──┬──────────────────────────────────┬───────────────────────────────────────────────────────►◄
   └─serverid=──server identifier─────┘
```

The following attributes are ignored:

- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST contable=ELACNxxx
```

Note: Specifying the CONTABLE attribute requires the three character NLS code appropriate to your language. See “Conversion Table by Language and Platform” on page 436 for more on information on specifying the CONTABLE attribute.

Configuring a Solaris Client for a VSECICS Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►─:calllink─applname=program name─linktype=─REMOTE─remotecomtype=─CICSCLIENT──────►

►─┬────────────────────────────────────────┬─┬────────────────────────────┬────────►
  └─contable=─┬─conversion table name─┬───┘ └─location=─┬─EZELOC──────┬─┘
              ├─'*' ──────────────────┤                 └─system name─┘
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬──────────────────────────┬─┬────────────────────────────┬──────────────────────►
  │            ┌─CLIENT─┐    │ └─parmform=─┬─OSLINK─────┬───┘
  └─luwcontrol=┴─SERVER─┴────┘             ├─COMMPTR────┤
                                           ├─COMMDATA───┤
                                           └─CICSOSLINK─┘

►─┬───────────────────────────┬─┬──────────────────────────────┬───────────────────►
  └─remoteapptype=─┬─VG────┬──┘ │            ┌─GENERATION─┐    │
                   ├─NONVG─┤    └─remotebind=┴─RUNTIME────┴────┘
                   └─ITF───┘

►─┬─────────────────────────────────┬──────────────────────────────────────────────►◄
  └─serverid=──server identifier───┘
```

The following attributes are ignored:
- externalname
- library

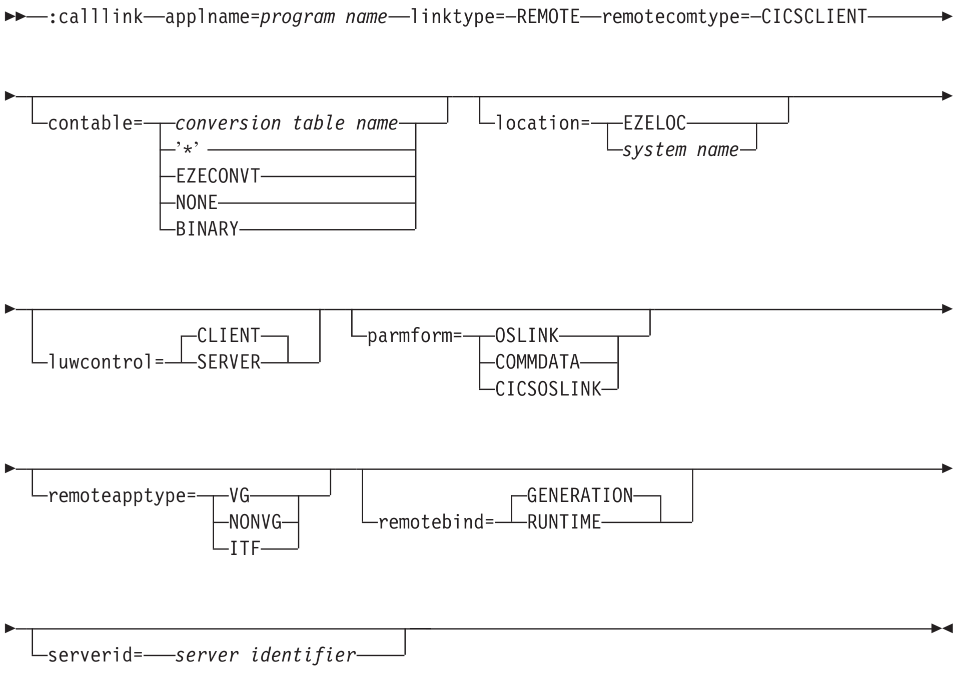
Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST contable=ELACNxxx
```

Note: Specifying the CONTABLE attribute requires the three character NLS code appropriate to your language. See “Conversion Table by Language and Platform” on page 436 for more on information on specifying the CONTABLE attribute.

Configuring a Solaris Client for a CICS for Solaris Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:



The following attributes are ignored:
- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST contable=BINARY
```

DIRECT Protocol

Advantages of IPC and DIRECT Protocols

The protocols IPC and DIRECT provide a method for client programs to call server programs where both reside on the same system. A local call from the client program to the server program might be used, but there are advantages to making this call remotely via the VisualAge Generator middleware in certain cases. Futhermore, these advantages differ for the provided protocols, IPC and DIRECT, and are described below.

Calls to local C++ applications for database access from Smalltalk GUI applications where the GUI application might run in the same session with other unrelated GUI applications that require database access: GUI applications begin VisualAge Generator communications sessions when calling server programs. When the IPC protocol is used, each call from a different GUI would be made under a different process, each of which can have its own database connection and unit of work. This would not be possible if the client program used a local call to access the server program.

If two GUI applications need to be in the same unit of work, start the second GUI from the first and define the second GUI with the VAGen inheritsCommSession attribute set to true in the AbtAppBldrPart or AbtAppBldrView class. This way, local servers called by both GUI applications will share the same communications session, database connection, and unit of work.

Calls to generated C++ application from Java clients where the C++ application is to run on a web server: Multiple calls might arrive simultaneously, thus the need for multiple, independent server-controlled units of work provided by the IPC protocol.

When using the DIRECT protocol, the call is similar to a local call directly from a client program to a server program except that the VisualAge Generator middleware performs the call. Because no separate process is created, the DIRECT protocol provides improved performance over the IPC protocol. The following are scenarios for using the DIRECT protocol:

Calls to a server program running under the control of VisualAge Generator test facility from Java, VisualBasic, or PowerBuilder, where the client and server resides on the same system.

Calls to a server program running under the control of Component Broker, which maintains the database connection itself. In such an environment, the server program should not run under a separate process so that the database connection cannot be inadvertently closed when the process ends.

Error Handling

The standard error handling procedures described in “Communication Error Handling” on page 22 are supported with IPC and DIRECT server calls. EZERT8 is set to the decimal value of the client access service reason code. The reason code is the same as the number portion of error message CSOnnnna as listed in the *VisualAge Generator Messages and Problem Determination Guide* document.

Configuring a Solaris Client for an AIX Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via DIRECT:

```
►►──:calllink──applname=program name──linktype=─CSOCALL──remotecomtype=─DIRECT──────────►◄
```

While the specification of:

```
►►──contable=─┬─conversion table name─┬──────────────────────────────────────────────────►◄
              ├─'*' ──────────────────┤
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘
```

is permissible, data conversion is generally not necessary when the client and server reside on the same system. All other linkage table entries are ignored on a local (DIRECT) call.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=CSOCALL remotecomtype=DIRECT
```

IPC Protocol

Advantages of IPC and DIRECT Protocols

The protocols IPC and DIRECT provide a method for client programs to call server programs where both reside on the same system. A local call from the client program to the server program might be used, but there are advantages to making this call remotely via the VisualAge Generator middleware in certain cases. Futhermore, these advantages differ for the provided protocols, IPC and DIRECT, and are described below.

Calls to local C++ applications for database access from Smalltalk GUI applications where the GUI application might run in the same session with other unrelated GUI applications that require database access: GUI applications begin VisualAge Generator communications sessions when calling server programs. When the IPC protocol is used, each call from a different GUI would be made under a different process, each of which can have its own database connection and unit of work. This would not be possible if the client program used a local call to access the server program.

If two GUI applications need to be in the same unit of work, start the second GUI from the first and define the second GUI with the VAGen inheritsCommSession attribute set to true in the AbtAppBldrPart or AbtAppBldrView class. This way, local servers called by both GUI applications will share the same communications session, database connection, and unit of work.

Calls to generated C++ application from Java clients where the C++ application is to run on a web server: Multiple calls might arrive simultaneously, thus the need for multiple, independent server-controlled units of work provided by the IPC protocol.

When using the DIRECT protocol, the call is similar to a local call directly from a client program to a server program except that the VisualAge Generator middleware performs the call. Because no separate process is created, the DIRECT protocol provides improved performance over the IPC protocol. The following are scenarios for using the DIRECT protocol:

Calls to a server program running under the control of VisualAge Generator test facility from Java, VisualBasic, or PowerBuilder, where the client and server resides on the same system.

Calls to a server program running under the control of Component Broker, which maintains the database connection itself. In such an environment, the server program should not run under a separate process so that the database connection cannot be inadvertently closed when the process ends.

Error Handling

The standard error handling procedures described in “Communication Error Handling” on page 22 are supported with IPC and DIRECT server calls. EZERT8 is set to the decimal value of the client access service reason code. The reason code is the same as the number portion of error message CSOnnnna as listed in the *VisualAge Generator Messages and Problem Determination Guide* document.

Configuring a Solaris Client for an AIX server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via IPC:

Configuring a Solaris Client

```
►►──:calllink──applname=program name──linktype=─CSOCALL──remotecomtype=─IPC──────────►◄
```

While the specification of:

```
►►──contable=─┬─conversion table name─┬──────────────────────────────────────────────►◄
              ├─'*'───────────────────┤
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘
```

is permissable, data conversion is generally not necessary when the client and server reside on the same system. All other linkage table entries are ignored on a local (IPC) call.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=CSOCALL remotecomtype=IPC
```

TCP/IP Protocol

Creating a TCP/IP Services File Entry

The client machine must have an entry for the TCP/IP Serverid added to its TCP/IP services file. For Windows, AIX, HP-UX, and Solaris the TCP/IP SERVICES file is used.

Error Handling

The standard error handling procedures described in "Communication Error Handling" on page 22 are supported with TCPIP server calls. EZERT8 is set to the decimal value of the client access service reason code. The reason code is the same as the number portion of error message CSOnnnna as listed in the *VisualAge Generator Messages and Problem Determination Guide* document.

Configuring a Solaris Client for an OS/2 server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─TCPIP──────────►
```

```
►─┬────────────────────────────────────┬──────────────────────────────►
  └─contable=─┬─conversion table name─┬┘
              ├─'*' ──────────────────┤
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬──────────────────────────────────────────┬────────────────────────►
  └─location=──system name_tcpip_hostname────┘

►─┬──────────────────────────────────────┬───────────────────────────►◄
  └─serverid=──tcpip_server identifier───┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to “Conversion Table by Language and Platform” on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 remotebind=RUNTIME serverid=port1
```

Configuring a Solaris client for a Windows NT server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►─:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─TCPIP──────►

►─┬────────────────────────────────────┬──────────────────────────────►
  └─contable=─┬─conversion table name─┬┘
              ├─'*' ──────────────────┤
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬──────────────────────────────────────────┬────────────────────────►
  └─location=──system name_tcpip_hostname────┘
```

```
►─┬──────────────────────────────────────┬──────────────────────────────►◄
  └─serverid=───tcpip_server identifier───┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to “Conversion Table by Language and Platform” on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 remotebind=RUNTIME serverid=port1
```

Configuring a Solaris Client for an AIX server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─TCPIP──────────►

►─┬────────────────────────────────────┬─────────────────────────────────────►
  └─contable=──┬─conversion table name─┬─┘
               ├─’*’ ──────────────────┤
               ├─EZECONVT──────────────┤
               ├─NONE──────────────────┤
               └─BINARY────────────────┘

►─┬──────────────────────────────────────────┬───────────────────────────────►
  └─location=───system name_tcpip_hostname───┘

►─┬──────────────────────────────────────┬───────────────────────────────►◄
  └─serverid=───tcpip_server identifier───┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to “Conversion Table by Language and Platform” on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 remotebind=RUNTIME serverid=port1
```

Configuring a Solaris client for an HP-UX server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►─:calllink─applname=program name─linktype=─REMOTE─remotecomtype=─TCPIP──────────►

►─┬─────────────────────────────────┬───────────────────────────────────────────►
  └─contable=─┬─conversion table name─┤
              ├─'*'─────────────────┤
              ├─EZECONVT────────────┤
              ├─NONE────────────────┤
              └─BINARY──────────────┘

►─┬─────────────────────────────────────────┬─────────────────────────────────►
  └─location=──system name_tcpip_hostname──┘

►─┬──────────────────────────────────────┬────────────────────────────────────►◄
  └─serverid=──tcpip_server identifier──┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to “Conversion Table by Language and Platform” on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 remotebind=RUNTIME serverid=port1
```

Configuring a Solaris Client for a Solaris server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►─:calllink─applname=program name─linktype=─REMOTE─remotecomtype=─TCPIP──────────►
```

```
►─┬─────────────────────────────────────┬──────────────────────────────────────►
  └─contable=─┬─conversion table name─┬─┘
              ├─'*' ──────────────────┤
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬───────────────────────────────────────────┬────────────────────────────────►
  └─location=──system name_tcpip_hostname─────┘

►─┬──────────────────────────────────────┬────────────────────────────────────►◄
  └─serverid=──tcpip_server identifier───┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to “Conversion Table by Language and Platform” on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 remotebind=RUNTIME serverid=port1
```

Configuring a Solaris Client for a VM/ESA server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─TCPIP───────────►

►─┬─────────────────────────────────────┬──────────────────────────────────────►
  └─contable=─┬─conversion table name─┬─┘
              ├─'*' ──────────────────┤
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬───────────────────────────────────────────┬────────────────────────────────►
  └─location=──system name_tcpip_hostname─────┘
```

```
►─┬─────────────────────────────────────┬───────────────────────────────►◄
  └─serverid=──tcpip_server identifier──┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to "Conversion Table by Language and Platform" on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 remotebind=RUNTIME serverid=port1
```

Configuring a Solaris Server

Summary Table of Valid Clients and Protocols

Table 22. Valid Clients and Protocols for VisualAge Generator Servers on the Solaris Platform

Server Platform	**Client Platforms**				
	OS/2 (GUI, C++, ITF)	**Windows 95 and Windows 98 (GUI)**	**Windows NT (GUI, C++, ITF)**	**AIX (C++)**	**Solaris (C++)**
Solaris	TCP/IP	TCP/IP	TCP/IP	TCP/IP	TCP/IP, IPC, DIRECT

DIRECT Protocol

List of Valid Clients

- Solaris (C++)

Advantages of IPC and DIRECT Protocols

The protocols IPC and DIRECT provide a method for client programs to call server programs where both reside on the same system. A local call from the client program to the server program might be used, but there are advantages to making this call remotely via the VisualAge Generator middleware in certain cases. Futhermore, these advantages differ for the provided protocols, IPC and DIRECT, and are described below.

Calls to local C++ applications for database access from Smalltalk GUI applications where the GUI application might run in the same session with other unrelated GUI applications that require database access: GUI applications begin VisualAge Generator communications sessions when calling server programs. When the IPC protocol is used, each call from a different GUI would be made under a different process, each of which can have its own database connection and unit of work. This would not be possible if the client program used a local call to access the server program.

If two GUI applications need to be in the same unit of work, start the second GUI from the first and define the second GUI with the VAGen inheritsCommSession attribute set to true in the AbtAppBldrPart or AbtAppBldrView class. This way, local servers called by both GUI applications will share the same communications session, database connection, and unit of work.

Calls to generated C++ application from Java clients where the C++ application is to run on a web server: Multiple calls might arrive simultaneously, thus the need for multiple, independent server-controlled units of work provided by the IPC protocol.

When using the DIRECT protocol, the call is similar to a local call directly from a client program to a server program except that the VisualAge Generator middleware performs the call. Because no separate process is created, the DIRECT protocol provides improved performance over the IPC protocol. The following are scenarios for using the DIRECT protocol:

Calls to a server program running under the control of VisualAge Generator test facility from Java, VisualBasic, or PowerBuilder, where the client and server resides on the same system.

Calls to a server program running under the control of Component Broker, which maintains the database connection itself. In such an environment, the server program should not run under a separate process so that the database connection cannot be inadvertently closed when the process ends.

Identifying a C++ Server Location

When using the IPC or DIRECT protocols, server programs are located using the LIBPATH environment variable. You need to ensure that the server DLL directory is specified in the LIBPATH.

IPC Protocol

List of Valid Clients

- Solaris (C++)

Advantages of IPC and DIRECT Protocols

The protocols IPC and DIRECT provide a method for client programs to call server programs where both reside on the same system. A local call from the client program to the server program might be used, but there are advantages to making this call remotely via the VisualAge Generator middleware in certain cases. Futhermore, these advantages differ for the provided protocols, IPC and DIRECT, and are described below.

Calls to local C++ applications for database access from Smalltalk GUI applications where the GUI application might run in the same session with other unrelated GUI applications that require database access: GUI applications begin VisualAge Generator communications sessions when calling server programs. When the IPC protocol is used, each call from a different GUI would be made under a different process, each of which can have its own database connection and unit of work. This would not be possible if the client program used a local call to access the server program.

If two GUI applications need to be in the same unit of work, start the second GUI from the first and define the second GUI with the VAGen inheritsCommSession attribute set to true in the AbtAppBldrPart or AbtAppBldrView class. This way, local servers called by both GUI applications will share the same communications session, database connection, and unit of work.

Calls to generated C++ application from Java clients where the C++ application is to run on a web server: Multiple calls might arrive simultaneously, thus the need for multiple, independent server-controlled units of work provided by the IPC protocol.

When using the DIRECT protocol, the call is similar to a local call directly from a client program to a server program except that the VisualAge Generator middleware performs the call. Because no separate process is created, the DIRECT protocol provides improved performance over the IPC protocol. The following are scenarios for using the DIRECT protocol:

Calls to a server program running under the control of VisualAge Generator test facility from Java, VisualBasic, or PowerBuilder, where the client and server resides on the same system.

Calls to a server program running under the control of Component Broker, which maintains the database connection itself. In such an environment, the server program should not run under a separate process so that the database connection cannot be inadvertently closed when the process ends.

Identifying the Server Location

When using the IPC or DIRECT protocols, server programs are located using the LIBPATH environment variable. You need to ensure that the server DLL directory is specified in the LIBPATH.

Configuring a Solaris Server

TCP/IP Protocol

List of Valid Clients

- OS/2 (GUI, C++, ITF)
- Windows 95 and Windows 98 (GUI)
- Windows NT (GUI, C++, ITF)
- AIX (C++)
- Solaris (C++)

Identifying a C++ Server Location

The TCP/IP support servers locate the server programs via the LIBPATH environment variable, so you need to ensure that the server DLL directory is specified in the LIBPATH.

Server Program Set Up and Operation

The server uses a configuration file for specifying the TCP/IP service name to listen on. The configuration file is optional as there are predefined default values that will be used.

Start the TCP/IP server by issuing the command:

```
CSOTCPS "config_filename"
```

The configuration file is located via the following search order:

1. File specified on the command line
2. The file named CSO.INI in the directory specified by the CSODIR environment variable.
3. The file named CSO.INI in the current directory

The search ends when the first one of the above conditions is met. Once a file is identified, the contents of the file are used if possible. If the file cannot be opened for any reason, a warning message is printed on the console and the default values are used. CSOTCPS will display the configuration filename and values being used prior to starting to listen for incoming requests. The default values are:

```
TCP/IP service name: VAGenerator
```

Where:

TCP/IP service name	Specifies the service name that will be used to look up the TCP/IP port number that CSOTCPS will listen on for incoming requests. This entry is case sensitive and must match an entry in the TCP/IP services file. For Windows NT, AIX, and HP-UX the TCP/IP SERVICES file is used.

Sample TCP/IP Entries from CSO.INI File: The following TCP/IP entries are located in the sample CSO.INI file. All entries must start in column 1. Comments are allowed within the file by placing a semicolon (;) in column 1 of the comment line. You only need to specify an entry if you wish to override the default value. If duplicate entries are specified in the file, the last entry in the file is used (just like environment variables in the config.sys file).

```
tcp_service_name=VAGenerator
```

Chapter 12. CICS for Solaris Platform

How to use this chapter

This chapter contains platform and protocol information necessary to configure a client program or a server program. The chapter is divided into configuration sections: client programs and server programs. Each configuration section is organized by protocol. Each protocol section contains information organized by target platforms.

Use the following method to locate the information you need:

1. Verify that this chapter is appropriate to your runtime environment.

   Ensure that the runtime environment of your program matches the platform on which this chapter is based. Turn to the appropriate chapter if the runtime environment of your program does not match the platform named in the title of this chapter.
2. Determine which type of program you want to configure: client program or server program.
3. Determine which configuration section in the chapter is appropriate for your program.
4. Find the intended target platform of the program you are configuring in the Summary Table of Valid Clients and Protocols or Summary Table of Valid Servers and Protocols.

   If you are configuring a client program, find the target platform of the intended server program in the column of valid platforms.

   If you are configuring a server program, find the target platform of the intended client program in the row of valid platforms.
5. Chose a protocol based on the information in the table.

   If you are configuring a client program, find the cell in the table where the row of the intended target platform intersects the column of the client platform. Select a protocol from the list of available protocols for this client/server combination.

   If you are configuring a client program, find the cell in the table where the column of the intended target platform intersects the row of the server platform. Select a protocol from the list of available protocols for this client/server combination.
6. Determine the name of the appropriate protocol section within the configuration section.

Turn to the part of the configuration section indicated by the protocol name. Here you will find information common to all target platforms valid for this protocol.

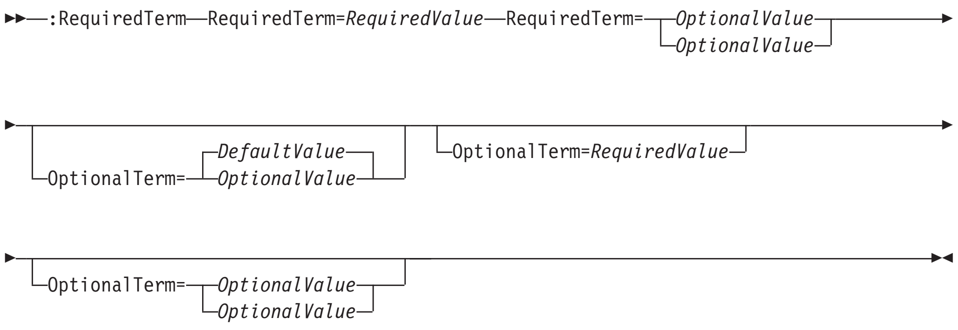
7. Determine the name of the appropriate target platform section within the protocol section.

   You will find platform specific information for the chosen protocol in the section named for your intended target platform.

Example

```
Chapter: CICS for Solaris Platform
Type of program to configure: Client program
Configuration section: Configuring a CICS for Solaris client
Intended target platform: CICS for MVS/ESA (i.e. a CICS for MVS/ESA server program)
Chosen protocol: CICS DPL
Protocol section: CICS DPL Protocol
Target platform section: Configuring a CICS for Solaris Client for a
CICS for MVS/ESA Server
```

You will encounter syntax diagrams in this chapter. The diagrams are used to show programming syntax as follows:



RequiredTerm	A required term is a term that must be specified. Required terms are depicted in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required terms.
RequiredValue	A required value is a value that must be specified for a term. A required value is shown in italics in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required values.
OptionalTerm	An optional term is a term that can be

	specified. An optional term is depicted in syntax diagrams in an alternate path from the preceding term.
OptionalValue	An optional value is a value that can be specified for a term. An optional value is shown in italics in syntax diagrams in an alternate path from the preceding term along with other optional values or a default value.
DefaultValue	A default value is the value that is specified by default for a term. A default value is shown in italics in syntax diagrams in an alternate path above other optional values.

Configuring a CICS for Solaris Client

Summary Table of Valid Servers and Protocols

Table 23. Valid Servers and Protocols for VisualAge Generator Clients on the CICS for Solaris Platform

Server Platforms	Protocols for CICS for Solaris Platform
CICS for AIX	CICS DPL
CICS for MVS/ESA	CICS DPL
CICS for OS/2	CICS DPL
CICS for Solaris	CICS DPL
CICS for VSE/ESA	CICS DPL
CICS for Windows NT	CICS DPL

CICS DPL Protocol

User Authentication

The user associated with the CICS client transaction must be authorized to run the client transaction and the server transaction. The user ID and password are identified by the CICS transaction manager on the client system.

Setting Up Communication Links

The communication link between the client and server systems must be defined to the CICS systems to allow a distributed program link from one system to the other. Refer to the CICS intercommunication manual for your CICS systems for more information.

Controlling the Unit of Work

Use the `luwcontrol` linkage table attribute to indicate whether the server updates are automatically committed on return or the client controls the unit of work. If the client is controlling the unit of work, subsequent server calls

must go to the same system and transaction under client-controlled unit of work until a commit or roll back is requested. The server cannot issue EZECOMIT or EZEROLLB calls if client-controlled unit of work is specified.

Data Format Conversion

Code format conversion is performed on the server call as specified in the `contable` linkage attribute table and in EZECONVT. Code is converted based on the structure of the arguments that are specified on the CALL statement in the client program.

Error Handling

The standard error handling procedures described below are supported with CICS DPL. If the DPL is not successful for any reason, including unavailability of the communication link, error information is returned with CICS in the CICS EIB (EXEC interface block).

If the REPLY option is not specified on the CALL statement, the calling program ends with error messages. If REPLY is specified, no messages are written or logged, and EZERT8 is set to one of the following values:

Value	**Meaning**
00000000	Successful call and return
00000204	Program name not valid
00000207	System identifier not valid
00000208	Link out of service
ffrrrrrr	Other CICS error where ff is the hexadecimal representation of EIBFN byte 0, and rrrrrr is the hexadecimal representation of EIBRCODE bytes 0–2.

Configuring a CICS for Solaris Client for a CICS for OS/2 Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►─:calllink─applname=program name─linktype=─REMOTE─┬────────────────────┬──────►
                                                    └─parmform=─COMMDATA─┘

►─┬─────────────────────────────────┬──────────────────────────────────────────►
  │             ┌─GENERATION─┐      │
  └─remotebind=─┴────────────┴─RUNTIME─┘
```

```
►─┬─────────────────────────────────────┬─┬──────────────────────────┬──────────►
  └─contable=─┬─conversion table name─┬─┘ └─location=─┬─EZELOC──────┬─┘
              ├─'*' ──────────────────┤               └─system name─┘
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬────────────────────────────┬─┬──────────────────────────────┬────────────────►
  │               ┌─CLIENT─┐   │ └─remoteapptype=─┬─VG────┬─────┘
  └─luwcontrol=───┴─SERVER─┴───┘                  ├─NONVG─┤
                                                  └─ITF───┘

►─┬─────────────────────────────────────┬────────────────────────────────────────►◄
  └─serverid=──server identifier────────┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Configuring a CICS for Solaris Client for a CICS for Windows NT Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►──:calllink──applname=program name──linktype=─REMOTE─┬──────────────────────┬──►
                                                       └─parmform=─COMMDATA───┘

►─┬──────────────────────────────────────────┬───────────────────────────────────►
  │              ┌─GENERATION─┐              │
  └─remotebind=──┴────────────┴──RUNTIME─────┘

►─┬─────────────────────────────────────┬─┬──────────────────────────┬──────────►
  └─contable=─┬─conversion table name─┬─┘ └─location=─┬─EZELOC──────┬─┘
              ├─'*' ──────────────────┤               └─system name─┘
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘
```

```
►─┬──────────────────────────────┬──┬───────────────────────────────┬──────────────►
  │              ┌─CLIENT─┐      │  └─remoteapptype=─┬─VG────┬──────┘
  └─luwcontrol=──┴─SERVER─┴──────┘                   ├─NONVG─┤
                                                     └─ITF───┘

►─┬────────────────────────────────┬───────────────────────────────────────────►◄
  └─serverid=──server identifier───┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Configuring a CICS for Solaris Client for a CICS for AIX Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►──:calllink──applname=program name──linktype=─REMOTE──┬──────────────────────┬──►
                                                        └─parmform=─COMMDATA───┘

►─┬──────────────────────────────────────────┬──────────────────────────────────►
  │              ┌─GENERATION─┐              │
  └─remotebind=──┴────────────┴──RUNTIME─────┘

►─┬──────────────────────────────────────┬──┬─────────────────────────────────┬──►
  └─contable=──┬─conversion table name─┬─┘  └─location=──┬─EZELOC──────┬─────┘
               ├─'*'───────────────────┤                 └─system name─┘
               ├─EZECONVT──────────────┤
               ├─NONE──────────────────┤
               └─BINARY────────────────┘

►─┬──────────────────────────────┬──┬───────────────────────────────┬──────────────►
  │              ┌─CLIENT─┐      │  └─remoteapptype=─┬─VG────┬──────┘
  └─luwcontrol=──┴─SERVER─┴──────┘                   ├─NONVG─┤
                                                     └─ITF───┘

►─┬────────────────────────────────┬───────────────────────────────────────────►◄
  └─serverid=──server identifier───┘
```

The following attributes are ignored:
- externalname

- library
- remotecomtype

Configuring a CICS for Solaris Client for a CICS for MVS/ESA Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►─:calllink─applname=program name─linktype=─REMOTE──┬─────────────────────┬──────►
                                                     └─parmform=─COMMDATA──┘

►─┬──────────────────────────────────────────┬────────────────────────────────────►
  │               ┌─GENERATION─┐             │
  └─remotebind=───┴────────────┴──RUNTIME────┘

►─┬──────────────────────────────────────────┬─┬──────────────────────────┬───────►
  └─contable=──┬─conversion table name──────┘ └─location=──┬─EZELOC──────┘
               ├─'*'────────────────────┤                  └─system name─┘
               ├─EZECONVT───────────────┤
               ├─NONE───────────────────┤
               └─BINARY─────────────────┘

►─┬─────────────────────────────┬─┬───────────────────────────────┬──────────────►
  │               ┌─CLIENT─┐    │ └─remoteapptype=──┬─VG────────┘
  └─luwcontrol=───┴─SERVER─┴────┘                   ├─NONVG─┤
                                                    └─ITF───┘

►─┬─────────────────────────────────────┬───────────────────────────────────────►◄
  └─serverid=──server identifier────────┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Configuring a CICS for Solaris Client for a CICS for VSE/ESA Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►─:calllink─applname=program name─linktype=─REMOTE──┬─────────────────────┬──────►
                                                     └─parmform=─COMMDATA──┘
```

Configuring a CICS for Solaris Client

```
►─┬───────────────────────────────────────────┬──────────────────────────────►
  │                 ┌─GENERATION─┐             │
  └─remotebind=─────┴────────────┴──RUNTIME────┘

►─┬────────────────────────────────────────┬─┬─────────────────────────────┬──►
  └─contable=──┬─conversion table name─┬───┘ └─location=──┬─EZELOC──────┬─┘
               ├─'*'───────────────────┤                  └─system name─┘
               ├─EZECONVT──────────────┤
               ├─NONE──────────────────┤
               └─BINARY────────────────┘

►─┬─────────────────────────────────┬─┬─────────────────────────────┬────────►
  │                 ┌─CLIENT─┐       │ └─remoteapptype=──┬─VG────┬───┘
  └─luwcontrol=─────┴─SERVER─┴───────┘                   ├─NONVG─┤
                                                          └─ITF───┘

►─┬──────────────────────────────────────┬───────────────────────────────────►◄
  └─serverid=──server identifier─────────┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Configuring a CICS for Solaris Server

Summary Table of Valid Clients and Protocols

Table 24. Valid Clients and Protocols for VisualAge Generator Servers on the CICS for Solaris Platform

Server Platform	**Client Platforms**					
	OS/2 (GUI, C++, ITF)	**Windows 95 and Windows 98 (GUI)**	**Windows NT (GUI, C++, ITF)**	**AIX (C++)**	**CICS**	**Solaris (C++)**
CICS for Solaris	CICS Client	CICS Client	CICS Client	CICS Client	CICS DPL	CICS Client

Notes:
CICS as a Client Platform refers to:
- CICS for AIX
- CICS for MVS/ESA
- CICS for OS/2
- CICS for Windows NT
- CICS for VSE/ESA
- CICS for Solaris

CICS Client Protocol

List of Valid Clients

- OS/2 (GUI, C++, ITF)
- Windows 95 and Windows 98 (GUI)
- Windows NT (GUI, C++, ITF)
- AIX (C++)
- Solaris (C++)

Identifying the Server Location

Specify the server location (system identifier) in the LOCATION linkage table attribute. You can also set the location dynamically in the client program at run time using EZELOC. If the server location is not specified, the default is the first entry in the CICS Client initialization file.

Linkage Table Attributes for Generating CICS Server Programs

Specify the following calllink attributes when you generate CICS server programs:

```
►►──:calllink──applname=program name──linktype=─REMOTE──┬────────────────────────┬──────►◄
                                                        └─parmform=─COMMDATA─────┘
```

The following attributes are ignored:

- contable
- externalname
- library
- location
- luwcontrol
- remoteapptype
- remotebind
- remotecomtype
- serverid

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA
```

Considerations for Defining the Server Program

If the remote program performs printing, it must move the print destination into EZEDESTP.

Server Program Set Up

The server program is generated and prepared as is any other CICS program on the server system. The server transaction must be defined to CICS on the server system. The default transaction name associated with a server program is the CICS-supplied mirror transaction, CPMI.

Specifying Parameter Format

Parameter format (the `parmform` attribute on the calllink tag) can be specified as COMMDATA (data values passed in the COMMAREA) or COMMPTR (pointers passed in the COMMAREA).

Using COMMPTR Format: Use COMMPTR for calling COBOL subroutines that were developed to use the non-VAGen parameter passing convention that uses pointers in the CICS COMMAREA. When the pointer format is used, the generated client program calls an intermediate catcher program (ELACSV, provided with VisualAge Generator host services) via the ECI instead of calling the server program directly. The catcher program receives the parameter data from the client, builds a pointer list in the COMMAREA that points to the individual parameters, and links to the server program. A program processing table (PPT) must be defined for the ELACSV program on the CICS server system.

COMMPTR format is only supported for MVS CICS, CICS for VSE/ESA, and CICS OS/2 servers.

Using COMMDATA Format: Use COMMDATA for calling generated programs and for calling COBOL subroutines where possible. COMMDATA provides better performance in calling remote programs. With COMMDATA, the server program is called directly without going through the catcher program.

CICS DPL Protocol

List of Valid Clients

- CICS for AIX
- CICS for MVS/ESA
- CICS for OS/2
- CICS for Windows NT
- CICS for VSE/ESA
- CICS for Solaris

Identifying the Server Location

You can specify the server location (CICS system identifier) on the `location` linkage table attribute, or dynamically set the location in the client program at runtime using EZELOC. If the location is not specified, the default location is the system identifier associated with the server program in the CICS program definition on the client system.

Linkage Table Attributes for Generating CICS Server Programs

Specify the following calllink attributes when you generate CICS server programs:

```
►►──:calllink──applname=program name──linktype=─REMOTE──┬──────────────────┬──────►◄
                                                        └─parmform=─COMMDATA─┘
```

The following attributes are ignored:
- contable
- externalname
- library
- location
- luwcontrol
- remoteapptype
- remotebind
- remotecomtype
- serverid

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA
```

Considerations for Defining the Server Program

If the remote program performs printing, it must move the print destination into EZEDESTP.

Server Program Set Up

The server program is generated and prepared as is any other CICS program on the server system. In addition, there must be a CICS program definition for the server program on the CICS client system containing the following information:
- Resident option = remote
- Remote system identifier
- Remote program name (optional)
- Remote transaction identifier (optional)

The server transaction must be defined to CICS on the server system. The default transaction name associated with a server program is the CICS-supplied mirror transaction, CPMI.

Chapter 13. VM/ESA Platform

How to use this chapter

This chapter contains platform and protocol information necessary to configure a client program or a server program. The chapter is divided into configuration sections: client programs and server programs. Each configuration section is organized by protocol. Each protocol section contains information organized by target platforms.

Use the following method to locate the information you need:

1. Verify that this chapter is appropriate to your runtime environment.

   Ensure that the runtime environment of your program matches the platform on which this chapter is based. Turn to the appropriate chapter if the runtime environment of your program does not match the platform named in the title of this chapter.
2. Determine which type of program you want to configure: client program or server program.
3. Determine which configuration section in the chapter is appropriate for your program.
4. Find the intended target platform of the program you are configuring in the Summary Table of Valid Clients and Protocols or Summary Table of Valid Servers and Protocols.

   If you are configuring a client program, find the target platform of the intended server program in the column of valid platforms.

   If you are configuring a server program, find the target platform of the intended client program in the row of valid platforms.
5. Chose a protocol based on the information in the table.

   If you are configuring a client program, find the cell in the table where the row of the intended target platform intersects the column of the client platform. Select a protocol from the list of available protocols for this client/server combination.

   If you are configuring a client program, find the cell in the table where the column of the intended target platform intersects the row of the server platform. Select a protocol from the list of available protocols for this client/server combination.
6. Determine the name of the appropriate protocol section within the configuration section.

Turn to the part of the configuration section indicated by the protocol name. Here you will find information common to all target platforms valid for this protocol.

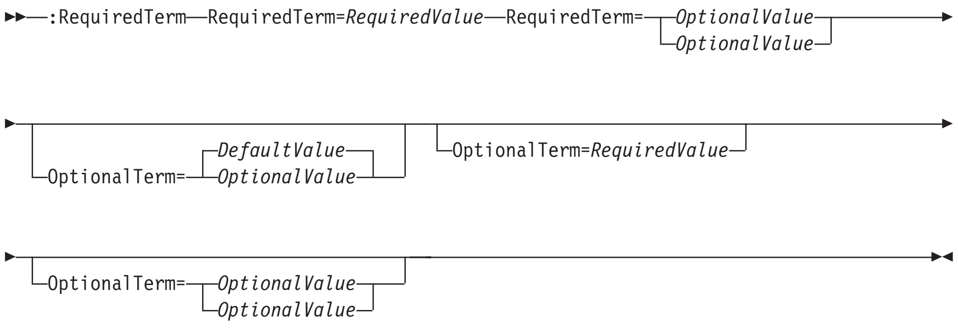
7. Determine the name of the appropriate target platform section within the protocol section.

   You will find platform specific information for the chosen protocol in the section named for your intended target platform.

Example

```
Chapter: VM/ESA Platform
Type of program to configure: Server program
Configuration section: Configuring a VM/ESA server
Intended target platform: OS/2 (i.e. an OS/2 client program)
Chosen protocol: DCE
Protocol section: DCE Protocol
```

You will encounter syntax diagrams in this chapter. The diagrams are used to show programming syntax as follows:



RequiredTerm	A required term is a term that must be specified. Required terms are depicted in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required terms.
RequiredValue	A required value is a value that must be specified for a term. A required value is shown in italics in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required values.
OptionalTerm	An optional term is a term that can be

	specified. An optional term is depicted in syntax diagrams in an alternate path from the preceding term.
OptionalValue	An optional value is a value that can be specified for a term. An optional value is shown in italics in syntax diagrams in an alternate path from the preceding term along with other optional values or a default value.
DefaultValue	A default value is the value that is specified by default for a term. A default value is shown in italics in syntax diagrams in an alternate path above other optional values.

Configuring a VM/ESA Client

Summary Table of Valid Servers and Protocols

Table 25. Valid Servers and Protocols for VisualAge Generator Clients on the VM/ESA Platform

Server Platforms	Protocols for VM/ESA Client Platform
AIX	N/A
CICS for AIX	N/A
HP-UX	N/A
IMS	N/A
CICS for MVS/ESA	N/A
OS/2	N/A
CICS for OS/2	N/A
OS/400	N/A
Solaris	N/A
CICS for Solaris	N/A
VM/ESA	N/A
CICS for VSE/ESA	N/A
Windows NT	N/A
CICS for Windows NT	N/A
Notes: N/A = Not available	

Note: The VM/ESA platform does not support VM/ESA clients.

Configuring a VM/ESA Server

Summary Table of Valid Clients and Protocols

Table 26. Valid Clients and Protocols for VisualAge Generator Servers on the VM/ESA Platform

Server Platform	Client Platforms				
	OS/2 (GUI, C++, ITF)	Windows 95 and Windows 98 (GUI)	Windows NT (GUI, C++, ITF)	AIX (C++)	Solaris (C++)
VM/ESA	TCP/IP	TCP/IP	TCP/IP	TCP/IP	TCP/IP

TCP/IP Protocol

List of Valid Clients

- OS/2 (GUI, C++, ITF)
- Windows 95 and Windows 98 (GUI)
- Windows NT (GUI, C++, ITF)
- AIX (C++)
- Solaris (C++)

Identifying a C++ Server Location

The TCP/IP support servers locate the server programs via the LIBPATH environment variable, so you need to ensure that the server DLL directory is specified in the LIBPATH.

Linkage Table Attributes for Generating Server Programs

A linkage table is not required for VM/ESA server programs. If you want to use a linkage table, you can use a linkage table for generating a client for VM/ESA as shown in the following example.

Example

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP location=server1
   remotebind=RUNTIME serverid=port1 contable=ELACNENU
```

Server Program Set Up and Operation

The VisualAge Generator Common Services TCP/IP server for VM is set up uniquely for the VM/CMS environment. Each "server" consists of at least one dispatcher virtual machine (DVM) and one or more application server virtual machines (AVMs).

The dispatcher virtual machine (DVM) must run as a disconnected service machine running the CSOTCPS MODULE (started by executing the CSODVM EXEC). The main function of the DVM is to listen for client calls and to dispatch incoming requests to an active application server virtual machine

(AVM). A startup list on the dispatcher virtual machine identifies the application server virtual machines to which incoming requests may be dispatched.

One or more AVMs must run as disconnected service machines running the CSOTCPW MODULE (started by executing the CSOAVM EXEC). The application server virtual machine accepts each client call passed to it by the DVM, executes the server application, and returns results to the client directly. Each AVM therefore needs access to the load library containing the generated server applications. An AVM accepts client calls from only one DVM identified by userid when the AVM is started.

The VisualAge Generator Common Services TCP/IP server for VM is designed to use the TCP/IP socket function of the underlying VM/ESA system. VM/ESA V2R2 or later is required. When setting up the DVM and AVM(s), ensure that the ″TCP/IP for VM″ product disk (often TCPMAINT 592) is not linked or accessible.

You need to set authority levels for DVM and AVM userids. For DVM userids:
- Set autolog authority for AVM userids
- Set IUCV ANY for communication with AVM userids
- Define a TCP/IP port in ETC SERVICES

For AVM userids, set IUCV ANY for communication with the DVM userid.

You can use your own naming convention for DVM and AVM userids. An example id for a DVM is CSOSERV. Example userids for AVMs associated with CSOSERV are CSOTCP1, CSOTCP2 and CSOTCP3.

Setting up the DVM on VM: Choose one or more virtual machines that will be dedicated to running CSOTCPS to listen for incoming client calls and dispatch the calls to AVMs. Large installations expecting high call volume may choose to run multiple DVMs but in most cases a single DVM is sufficient. If more than one dispatcher virtual machine is to be used, ensure that the TCP/IP Service Name specified in the configuration file or, by default, ″VAGenerator″ is unique for each. This service name must also be defined in the ETC SERVICES file for your VM system.

The DVM requires a startup list and optionally a configuration file. The contents of the startup list and configuration file are described below.

The file identifier for the startup list may be set in the configuration file or, by default, is set to ″CSOSTART LIST *″. This file is required because it identifies to the DVM the names of the AVMs to which it can dispatch incoming requests. A minimum of one AVM in the startup list is required.

The file identifier for the configuration file may be set when customizing CSODVM EXEC as described below. The default file identifier for the configuration file is ″CSO INI *″. This file is optional as there are predefined defaults that will be used when no configuration file is provided or when the provided configuration file cannot be opened.

The DVM is started by executing the CSODVM EXEC with no parameters:

```
CSODVM
```

The CSODVM EXEC requires the following customization:

1. Autologging of AVMs may optionally be done in the CSODVM EXEC. If you choose to do so, be sure that the dispatcher virtual machine has the proper VM authority to autolog the AVMs.
2. The environment variable, CSOTROPT, is set here to control tracing level. By default, the CSODVM EXEC sets CSOTROPT=1 which traces errors only. For more tracing, CSOTROPT may be set to 2.
3. The environment variable, CSONLS, is set here to control national language used for messages. By default, the CSODVM EXEC sets CSONLS=ENU which produces English-only messages. The other valid settings for CSONLS are: CHS, DES, DEU, ENP, ESP, JPN, KOR, and PTB.
4. CSODVM EXEC must link to VM minidisks containing the LE runtime code and the VisualAge Generator product. The link information must be modified to use the correct link locations specific to your VM system. Do NOT link to any disk containing the TCP/IP for VM product (TCPMAINT 592).
5. CSOTCPS MODULE is invoked by CSODVM EXEC. The module optionally accepts as parameters the ″filename filetype filemode″ of the configuration file. If not specified, the default is ″CSO INI *″.

When CSODVM EXEC is executed, the server will display the configuration values being used prior to starting to listen for incoming requests. The default values are:

```
TCP/IP Service Name:  VAGenerator
TCP/IP Startup List:  CSOSTART LIST *
```

TCP/IP Service Name

″vmtcp_service_name=″ is the tag in the configuration file used to specify the TCP/IP Service Name.

This is the service name that will be used to look up the TCP/IP port number that CSOTCPS will listen on for incoming requests. This entry is case sensitive and must match an entry in the TCP/IP ″ETC SERVICES″ file.

TCP/IP Startup List

″vmtcp_startup_list=″ is the tag in the configuration file used to specify the file identifier for the TCP/IP startup list.

The startup list identifies the AVMs that will accept incoming requests from this DVM. Optionally, it may contain control information to restrict requests from specific client IP addresses to one or more specific AVMs.

The AVMs provide a similar capability of restricting applications by the IP address of the requesting client. Refer to the discussion of setting up the AVMs below for details.

Because each of these control mechanisms relies on restriction by client IP address, they are designed to provide some measure of control but should be considered only as secure as the client IP address itself.

Contents and Format of the Configuration File: The configuration file entries must start in column 1. Comments are allowed within the file by placing a semicolon (;) in column 1 of the comment line. You only need to specify an entry if you wish to override the default value. If duplicate entries are specified in the file, the last entry in the file is used.

```
vmtcp_service_name=VAGenerator
vmtcp_startup_list=CSOSTART LIST *
```

Contents and Format of the Startup List: The startup file entries must start in column 1. Comments are allowed by placing a semicolon (;) in column 1 of the comment line. You may also use ″in-line″ comments placed to the right side only, as shown in the example below.

The startup list is used to identify all AVMs that will accept requests from this DVM. The AVMs will communicate with the DVM and will be used to service the requests that arrive from clients. Any AVMs that may be used should be specified here although there is a small performance penalty for including any that may not actually be started. The DVM will not be able to communicate with any AVMs that do not appear in the startup list. A minimum of one AVM must be specified. AVMs are identified in the startup list by use of the tag :AVM in columns 1-4 and followed, on the same line, by the userid of the AVM.

Optionally, incoming client requests may be restricted by client IP address to run on specific AVMs. Each client IP address which is to be restricted as such must be identified in the startup list. The identification of restricted client IP addresses is done by use of the tag :IP in columns 1-3 and followed, on the same line, by the client IP address in dotted decimal format. If a particular client IP address does not appear in the list, it may run on any available AVM. All non-comment lines following the :IP tagged line are presumed to be the userids of the AVMs that can accept incoming requests originating from that client IP address. Note that if an :IP tag line appears where no AVMs are named following it, it is ignored and requests from that client IP address can run on any AVM. Any invalid client IP addresses specified on an :IP tag are ignored as well. If a particular client IP address appears on more than one :IP tagged line, only the last occurrence is used.

```
; Sample Startup List
; Three AVMs are identified: CSOTCP1, CSOTCP2, and CSOTCP3
;
:AVM CSOTCP1
:AVM CSOTCP2
:AVM CSOTCP3  * In-line comments to the right are OK!
;
; Client IP address for Mary is restricted to only 1 AVM
;
:IP 9.35.135.03  * Follow by only 1 AVM she is restricted to
   CSOTCP2       * In-line comments here are ok too.
;
; Client IP address for Joe is restricted to either of
; two AVMs: CSOTCP1 or CSOTCP3.
;
:IP 9.35.143.02
   CSOTCP1
   CSOTCP3
```

Setting up AVMs on VM: An AVM is started by running the CSOAVM EXEC. Optionally, a control file named CSOAVM CONTROL may be used to restrict access to applications by IP address of the requesting client. If the control file named CSOAVM CONTROL exists and can be opened successfully, any application controls specified within are used. If the file does not exist or cannot be opened, no application controls will be active. This control mechanism is designed to provide some measure of security but can be considered only as secure as the client IP address itself.

An application load library containing the generated server applications must be accessible to each AVM. Server applications must be generated for target system VMCMS. The CSOAVM EXEC contains a GLOBAL LOADLIB command which must be customized to include the name of your application load library that is to be used by the AVM.

Format of the control file, CSOAVM CONTROL, is described below.

The CSOAVM EXEC requires the following customization:

1. The environment variable, CSOTROPT, is set here to control tracing level. By default, the CSOAVM EXEC sets CSOTROPT=1 which traces errors only. For more tracing, CSOTROPT may be set to 2.
2. The environment variable, CSO_DUMP_DATA, is set here to control dumping of argument data before and after the call. By default, CSO_DUMP_DATA=0, thus dumping of argument data is suppressed.
3. The environment variable, CSONLS, is set here to control national language used for messages. By default, the CSOAVM EXEC sets

CSONLS=ENU which produces English-only messages. The other valid settings for CSONLS are: CHS, DES, DEU, ENP, ESP, JPN, KOR, and PTB.

4. CSOAVM EXEC must link to VM minidisks containing the LE runtime code and the VisualAge Generator product. The link information must be modified to use the correct link locations specific to your VM system. Do NOT link to any disk containing the TCP/IP for VM product (TCPMAINT 592).
5. CSOAVM EXEC must run ELASETUP.
6. CSOAVM EXEC must be issue GLOBAL LOADLIB for the load libraries needed by the VisualAge Generator product and by your application.
7. CSOTCPW MODULE is invoked by CSOAVM EXEC. The format for invocation is:

   ```
   CSOTCPW [[dvm_userid] this_avm_userid]
   ```

   Where dvm_userid defaults to ″CSOTCPS″ and this_avm_userid defaults to ″UNKNOWN″. The sample exec provided queries the userid of the AVM & places the value as the 2nd parameter thus avoiding hard-coding of AVM userid on the call. If your DVM userid is not CSOTCPS, be sure to specify the correct DVM userid as the first parameter.

Contents and Format of the CSOAVM Control File: Control file entries must start in column 1. Comments are allowed by placing a semicolon (;) in column 1 of the comment line. You may also use ″in-line″ comments placed to the right side only, as shown in the example below.

Controlled applications are identified by a :APPL tag beginning in column 1 and immediately followed, on the same line, by the application name. Applications may be restricted by client IP address of the incoming client requests. Any application which is not restricted in this file may be run by any incoming client request. To restrict an application, include an :APPL tag immediately followed, on the same line, by the application name. All non-comment lines following the :APPL tag are presumed to be client IP addresses that are permitted to run that application. Note that if an :APPL tag appears with no client IP addresses that follow it before the next :APPL tag or before end-of-file, that application is not restricted. Further, if an application appears more than once in the file, only the last occurrence is used.

```
; Sample CSOAVM CONTROL file.
; Application PAYROLL is restricted to a single client IP
;
:APPL PAYROLL    * Follow by only Mary's client ip address
   9.35.135.03   * In-line comments here are ok too.
;
; Only two IP client addresses may run the QUERY application
; Note that no other applications are restricted.  They may be
; run by any client ip address.
;
```

```
:APPL QUERY
   9.35.135.03
   9.35.143.02   * Mary and Joe can run the QUERY application
```

In summary, the required files for the DVM are:

- CSOTCPS MODULE
- CSODVM EXEC
- CSOxxx JAVAMSG - one per language
- startup file - ″CSOSTART LIST″ by default of name specified in configuration file

The optional file for the DVM is the configuration file - ″CSO INI″ by default or name specified in CSODVM EXEC

The required files for the AVM are:

- CSOTCPW MODULE
- CSOAVM EXEC
- CSOxxx JAVAMSG - one per language

The optional file for the AVM is CSOAVM CONTROL

Chapter 14. CICS for VSE/ESA Platform

How to use this chapter

This chapter contains platform and protocol information necessary to configure a client program or a server program. The chapter is divided into configuration sections: client programs and server programs. Each configuration section is organized by protocol. Each protocol section contains information organized by target platforms.

Use the following method to locate the information you need:

1. Verify that this chapter is appropriate to your runtime environment.

   Ensure that the runtime environment of your program matches the platform on which this chapter is based. Turn to the appropriate chapter if the runtime environment of your program does not match the platform named in the title of this chapter.
2. Determine which type of program you want to configure: client program or server program.
3. Determine which configuration section in the chapter is appropriate for your program.
4. Find the intended target platform of the program you are configuring in the Summary Table of Valid Clients and Protocols or Summary Table of Valid Servers and Protocols.

   If you are configuring a client program, find the target platform of the intended server program in the column of valid platforms.

   If you are configuring a server program, find the target platform of the intended client program in the row of valid platforms.
5. Chose a protocol based on the information in the table.

   If you are configuring a client program, find the cell in the table where the row of the intended target platform intersects the column of the client platform. Select a protocol from the list of available protocols for this client/server combination.

   If you are configuring a client program, find the cell in the table where the column of the intended target platform intersects the row of the server platform. Select a protocol from the list of available protocols for this client/server combination.
6. Determine the name of the appropriate protocol section within the configuration section.

Turn to the part of the configuration section indicated by the protocol name. Here you will find information common to all target platforms valid for this protocol.

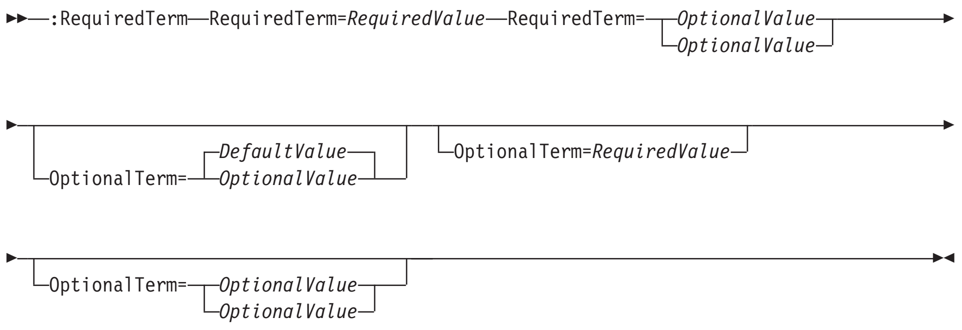
7. Determine the name of the appropriate target platform section within the protocol section.

   You will find platform specific information for the chosen protocol in the section named for your intended target platform.

Example

```
Chapter: CICS for VSE/ESA Platform
Type of program to configure: Client program
Configuration section: Configuring an CICS for VSE/ESA client
Intended target platform: CICS for MVS/ESA (i.e. a CICS for MVS/ESA server program)
Chosen protocol: CICS DPL
Protocol section: CICS DPL Protocol
Target platform section: Configuring a CICS for VSE/ESA Client for a
CICS for MVS/ESA Server
```

You will encounter syntax diagrams in this chapter. The diagrams are used to show programming syntax as follows:



RequiredTerm	A required term is a term that must be specified. Required terms are depicted in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required terms.
RequiredValue	A required value is a value that must be specified for a term. A required value is shown in italics in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required values.
OptionalTerm	An optional term is a term that can be

	specified. An optional term is depicted in syntax diagrams in an alternate path from the preceding term.
OptionalValue	An optional value is a value that can be specified for a term. An optional value is shown in italics in syntax diagrams in an alternate path from the preceding term along with other optional values or a default value.
DefaultValue	A default value is the value that is specified by default for a term. A default value is shown in italics in syntax diagrams in an alternate path above other optional values.

Configuring a CICS for VSE/ESA Client

Summary Table of Valid Servers and Protocols

Table 27. Valid Servers and Protocols for VisualAge Generator Clients on the CICS for VSE/ESA Platform

Server Platforms	Protocols for CICS for VSE/ESA Client Platform
CICS for AIX	CICS DPL
CICS for MVS/ESA	CICS DPL
CICS for OS/2	CICS DPL
CICS for Solaris	CICS DPL
CICS for VSE/ESA	CICS DPL
CICS for Windows NT	CICS DPL

CICS DPL Protocol

User Authentication

The user associated with the CICS client transaction must be authorized to run the client transaction and the server transaction. The user ID and password are identified by the CICS transaction manager on the client system.

Setting Up Communication Links

The communication link between the client and server systems must be defined to the CICS systems to allow a distributed program link from one system to the other. Refer to the CICS intercommunication manual for your CICS systems for more information.

Controlling the Unit of Work

Use the `luwcontrol` linkage table attribute to indicate whether the server updates are automatically committed on return or the client controls the unit of work. If the client is controlling the unit of work, subsequent server calls must go to the same system and transaction under client-controlled unit of work until a commit or roll back is requested. The server cannot issue EZECOMIT or EZEROLLB calls if client-controlled unit of work is specified.

Data Format Conversion

Code format conversion is performed on the server call as specified in the `contable` linkage attribute table and in EZECONVT. Code is converted based on the structure of the arguments that are specified on the CALL statement in the client program.

Error Handling

The standard error handling procedures described below are supported with CICS DPL. If the DPL is not successful for any reason, including unavailability of the communication link, error information is returned with CICS in the CICS EIB (EXEC interface block).

If the REPLY option is not specified on the CALL statement, the calling program ends with error messages. If REPLY is specified, no messages are written or logged, and EZERT8 is set to one of the following values:

Value	Meaning
00000000	Successful call and return
00000204	Program name not valid
00000207	System identifier not valid
00000208	Link out of service
ffrrrrrr	Other CICS error where ff is the hexadecimal representation of EIBFN byte 0, and rrrrrr is the hexadecimal representation of EIBRCODE bytes 0–2.

Configuring a CICS for VSE/ESA Client for a CICS for OS/2 Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►─:calllink─applname=program name─linktype=─REMOTE─┬──────────────────────┬───────►
                                                    └─parmform=─COMMDATA───┘
```

```
►─┬──────────────────────────────────────┬──────────────────────────────────────────►
  │              ┌─GENERATION─┐          │
  └─remotebind=──┴────────────┴─RUNTIME──┘


►─┬─────────────────────────────────────────┬─┬──────────────────────────────┬──────►
  └─contable=──┬─conversion table name─┬────┘ └─location=──┬─EZELOC──────┬──┘
               ├─'*' ──────────────────┤                   └─system name─┘
               ├─EZECONVT──────────────┤
               ├─NONE──────────────────┤
               └─BINARY────────────────┘


►─┬────────────────────────────┬─┬────────────────────────────────┬─────────────────►
  │              ┌─CLIENT─┐    │ └─remoteapptype=──┬─VG────┬──────┘
  └─luwcontrol=──┴─SERVER─┴────┘                   ├─NONVG─┤
                                                   └─ITF───┘


►─┬────────────────────────────────────┬──────────────────────────────────────────►◄
  └─serverid=──server identifier───────┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Example Linkage Table

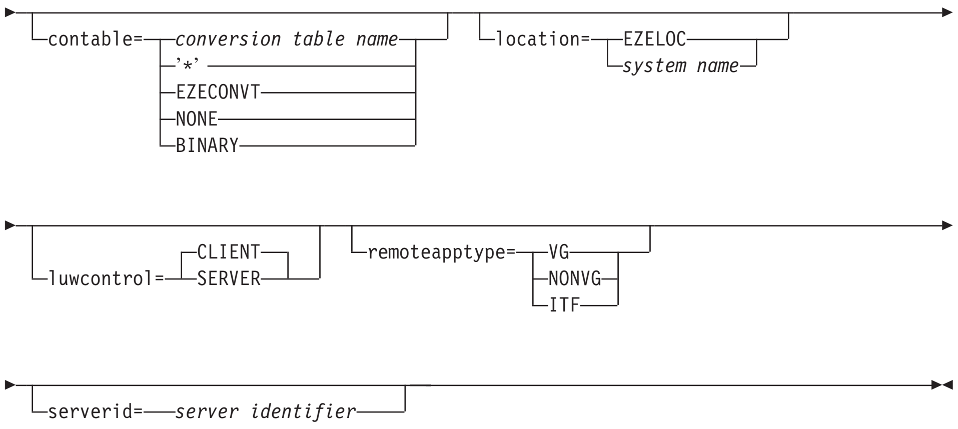
```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CS07 contable=ELACNENU LUWCONTROL=CLIENT
```

Configuring a CICS for VSE/ESA Client for a CICS for Windows NT Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►──:calllink──applname=program name──linktype=─REMOTE──┬─────────────────────┬────►
                                                        └─parmform=─COMMDATA──┘


►─┬──────────────────────────────────────┬──────────────────────────────────────────►
  │              ┌─GENERATION─┐          │
  └─remotebind=──┴────────────┴─RUNTIME──┘
```



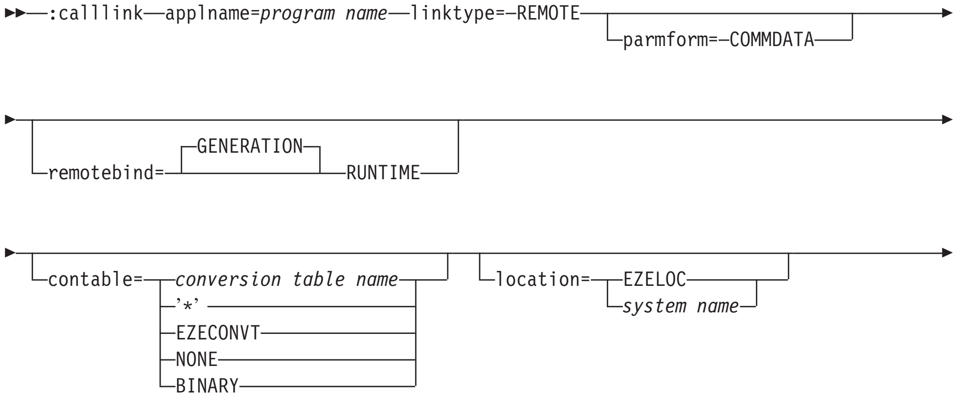
The following attributes are ignored:
- externalname
- library
- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CS07 contable=ELACNENU LUWCONTROL=CLIENT
```

Configuring a CICS for VSE/ESA Client for a CICS for AIX Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:



```
►─┬──────────────────────────┬─┬─────────────────────────┬──────────────────►
  │              ┌─CLIENT─┐  │ └─remoteapptype=─┬─VG────┬┘
  └─luwcontrol=──┴─SERVER─┴──┘                  ├─NONVG─┤
                                                └─ITF───┘

►─┬─────────────────────────────────┬────────────────────────────────────────►◄
  └─serverid=──server identifier────┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CS07 contable=ELACNENU LUWCONTROL=CLIENT
```

Configuring a CICS for VSE/ESA Client for a CICS for MVS/ESA Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►──:calllink──applname=program name──linktype=─REMOTE──┬─────────────────────┬──►
                                                        └─parmform=─COMMDATA──┘

►─┬────────────────────────────────────────┬─────────────────────────────────►
  │             ┌─GENERATION─┐             │
  └─remotebind=─┴────────────┴──RUNTIME────┘

►─┬───────────────────────────────────────┬─┬──────────────────────────────┬──►
  └─contable=─┬─conversion table name─────┬┘ └─location=─┬─EZELOC──────┬──┘
              ├─'*' ──────────────────────┤              └─system name─┘
              ├─EZECONVT──────────────────┤
              ├─NONE──────────────────────┤
              └─BINARY────────────────────┘

►─┬──────────────────────────┬─┬─────────────────────────┬──────────────────►
  │              ┌─CLIENT─┐  │ └─remoteapptype=─┬─VG────┬┘
  └─luwcontrol=──┴─SERVER─┴──┘                  ├─NONVG─┤
                                                └─ITF───┘
```

```
►─┬──────────────────────────────┬───────────────────────────────────────────────►◄
  └─serverid=──server identifier─┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CS07 contable=ELACNENU LUWCONTROL=CLIENT
```

Configuring a CICS for VSE/ESA Client for a CICS for VSE/ESAServer

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►──:calllink──applname=program name──linktype=─REMOTE─┬────────────────────┬─────►
                                                        └─parmform=─COMMDATA─┘

►─┬──────────────────────────────────────┬─────────────────────────────────────────►
  │                ┌─GENERATION─┐        │
  └─remotebind=────┴────────────┴─RUNTIME┘

►─┬──────────────────────────────────────┬─┬───────────────────────────┬───────────►
  └─contable=─┬─conversion table name─┬──┘ └─location=─┬─EZELOC──────┬─┘
              ├─'*' ──────────────────┤                └─system name─┘
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬───────────────────────────┬─┬────────────────────────────┬─────────────────────►
  │               ┌─CLIENT─┐  │ └─remoteapptype=─┬─VG────┬───┘
  └─luwcontrol=───┴─SERVER─┴──┘                  ├─NONVG─┤
                                                 └─ITF───┘

►─┬──────────────────────────────┬───────────────────────────────────────────────►◄
  └─serverid=──server identifier─┘
```

The following attributes are ignored:
- externalname
- library

- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CS07 contable=ELACNENU LUWCONTROL=CLIENT
```

Configuring a CICS for VSE/ESA Client for a CICS for Solaris Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►─:calllink─applname=program name─linktype=─REMOTE─┬─────────────────────┬──────►
                                                    └─parmform=─COMMDATA──┘

►─┬─────────────────────────────────────────┬────────────────────────────────────►
  │               ┌─GENERATION─┐            │
  └─remotebind=───┴────────────┴──RUNTIME───┘

►─┬───────────────────────────────────────────┬─┬───────────────────────────────┬─►
  └─contable=──┬─conversion table name──┬─────┘ └─location=──┬─EZELOC──────┬───┘
               ├─'*' ───────────────────┤                    └─system name─┘
               ├─EZECONVT───────────────┤
               ├─NONE───────────────────┤
               └─BINARY─────────────────┘

►─┬─────────────────────────────┬─┬───────────────────────────────┬──────────────►
  │              ┌─CLIENT──┐    │ └─remoteapptype=──┬─VG────┬─────┘
  └─luwcontrol=──┴─SERVER──┴────┘                   ├─NONVG─┤
                                                    └─ITF───┘

►─┬──────────────────────────────────┬──────────────────────────────────────────►◄
  └─serverid=───server identifier────┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CS07 contable=ELACNENU LUWCONTROL=CLIENT
```

Configuring a CICS for VSE/ESA Server

Summary Table of Valid Clients and Protocols

Table 28. Valid Clients and Protocols for VisualAge Generator Servers on the CICS for VSE/ESA Platform

Server Platform	**Client Platforms**					
	OS/2 (GUI, C++, ITF)	**Windows 95 and Windows 98 (GUI)**	**Windows NT (GUI, C++, ITF)**	**AIX (C++)**	**CICS**	**Solaris (C++)**
CICS for VSE/ESA	CICS Client	CICS Client	CICS Client	CICS Client	CICS DPL	CICS Client

Notes:
CICS as a Client Platform refers to:
- CICS for AIX
- CICS for MVS/ESA
- CICS for OS/2
- CICS for Windows NT
- CICS for VSE/ESA
- CICS for Solaris

CICS Client Protocol

List of Valid Clients

- OS/2 (GUI, C++, ITF)
- Windows 95 and Windows 98 (GUI)
- Windows NT (GUI, C++, ITF)
- AIX (C++)
- Solaris (C++)

Identifying the Server Location

Specify the server location (system identifier) in the LOCATION linkage table attribute. You can also set the location dynamically in the client program at run time using EZELOC. If the server location is not specified, the default is the first entry in the CICS Client initialization file.

Linkage Table Attributes for Generating CICS Server Programs

Specify the following calllink attributes when you generate CICS server programs:

```
►►─:calllink─applname=program name─linktype=─REMOTE──┬──────────────────────┬──────►◄
                                                     └─parmform=─COMMDATA───┘
```

The following attributes are ignored:
- contable

- externalname
- library
- location
- luwcontrol
- remoteapptype
- remotebind
- remotecomtype
- serverid

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA
```

Considerations for Defining the Server Program

If the remote program performs printing, it must move the print destination into EZEDESTP.

Server Program Set Up

The server program is generated and prepared as is any other CICS program on the server system. The server transaction must be defined to CICS on the server system. The default transaction name associated with a server program is the CICS-supplied mirror transaction, CPMI.

Specifying Parameter Format

Parameter format (the `parmform` attribute on the calllink tag) can be specified as COMMDATA (data values passed in the COMMAREA) or COMMPTR (pointers passed in the COMMAREA).

Using COMMPTR Format: Use COMMPTR for calling COBOL subroutines that were developed to use the non-VAGen parameter passing convention that uses pointers in the CICS COMMAREA. When the pointer format is used, the generated client program calls an intermediate catcher program (ELACSV, provided with VisualAge Generator host services) via the ECI instead of calling the server program directly. The catcher program receives the parameter data from the client, builds a pointer list in the COMMAREA that points to the individual parameters, and links to the server program. A program processing table (PPT) must be defined for the ELACSV program on the CICS server system.

COMMPTR format is only supported for MVS CICS, CICS for VSE/ESA, and CICS OS/2 servers.

Using COMMDATA Format: Use COMMDATA for calling generated programs and for calling COBOL subroutines where possible. COMMDATA provides better performance in calling remote programs. With COMMDATA, the server program is called directly without going through the catcher program.

Configuring a CICS for VSE/ESA Server

CICS DPL Protocol

List of Valid Clients
- CICS for AIX
- CICS for MVS/ESA
- CICS for OS/2
- CICS for Windows NT
- CICS for VSE/ESA
- CICS for Solaris

Identifying the Server Location
You can specify the server location (CICS system identifier) on the `location` linkage table attribute, or dynamically set the location in the client program at runtime using EZELOC. If the location is not specified, the default location is the system identifier associated with the server program in the CICS program definition on the client system.

Linkage Table Attributes for Generating CICS Server Programs
Specify the following calllink attributes when you generate CICS server programs:

```
►►──:calllink──applname=program name──linktype=─REMOTE──┬──────────────────────┬──────►◄
                                                         └─parmform=─COMMDATA───┘
```

The following attributes are ignored:
- contable
- externalname
- library
- location
- luwcontrol
- remoteapptype
- remotebind
- remotecomtype
- serverid

Example Linkage Table
```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA
```

Considerations for Defining the Server Program
If the remote program performs printing, it must move the print destination into EZEDESTP.

Server Program Set Up

The server program is generated and prepared as is any other CICS program on the server system. In addition, there must be a CICS program definition for the server program on the CICS client system containing the following information:

- Resident option = remote
- Remote system identifier
- Remote program name (optional)
- Remote transaction identifier (optional)

The server transaction must be defined to CICS on the server system. The default transaction name associated with a server program is the CICS-supplied mirror transaction, CPMI.

Chapter 15. Windows 95 and Windows 98 Platform

How to use this chapter

This chapter contains platform and protocol information necessary to configure a client program or a server program. The chapter is divided into configuration sections: client programs and server programs. Each configuration section is organized by protocol. Each protocol section contains information organized by target platforms.

Use the following method to locate the information you need:

1. Verify that this chapter is appropriate to your runtime environment.

   Ensure that the runtime environment of your program matches the platform on which this chapter is based. Turn to the appropriate chapter if the runtime environment of your program does not match the platform named in the title of this chapter.
2. Determine which type of program you want to configure: client program or server program.
3. Determine which configuration section in the chapter is appropriate for your program.
4. Find the intended target platform of the program you are configuring in the Summary Table of Valid Clients and Protocols or Summary Table of Valid Servers and Protocols.

   If you are configuring a client program, find the target platform of the intended server program in the column of valid platforms.

   If you are configuring a server program, find the target platform of the intended client program in the row of valid platforms.
5. Chose a protocol based on the information in the table.

   If you are configuring a client program, find the cell in the table where the row of the intended target platform intersects the column of the client platform. Select a protocol from the list of available protocols for this client/server combination.

   If you are configuring a client program, find the cell in the table where the column of the intended target platform intersects the row of the server platform. Select a protocol from the list of available protocols for this client/server combination.
6. Determine the name of the appropriate protocol section within the configuration section.

Turn to the part of the configuration section indicated by the protocol name. Here you will find information common to all target platforms valid for this protocol.

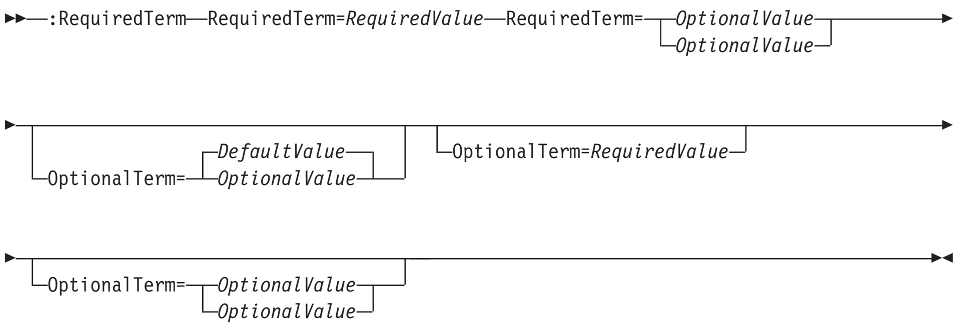
7. Determine the name of the appropriate target platform section within the protocol section.

   You will find platform specific information for the chosen protocol in the section named for your intended target platform.

Example

```
Chapter: Windows 95 and Windows 98 Platform
Type of program to configure: Client program
Configuration section: Configuring a Windows 95 and Windows 98 client
Intended target platform: CICS for MVS/ESA (i.e. a CICS for MVS/ESA server program)
Chosen protocol: CICS Client
Protocol section: CICS Client Protocol
Target platform section: Configuring a Windows 95 and Windows 98 Client
for a CICS for MVS/ESA Server
```

You will encounter syntax diagrams in this chapter. The diagrams are used to show programming syntax as follows:



RequiredTerm	A required term is a term that must be specified. Required terms are depicted in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required terms.
RequiredValue	A required value is a value that must be specified for a term. A required value is shown in italics in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required values.
OptionalTerm	An optional term is a term that can be

	specified. An optional term is depicted in syntax diagrams in an alternate path from the preceding term.
OptionalValue	An optional value is a value that can be specified for a term. An optional value is shown in italics in syntax diagrams in an alternate path from the preceding term along with other optional values or a default value.
DefaultValue	A default value is the value that is specified by default for a term. A default value is shown in italics in syntax diagrams in an alternate path above other optional values.

Configuring a Windows 95 and Windows 98 Client

Summary Table of Valid Servers and Protocols

Table 29. Valid Servers and Protocols for VisualAge Generator Clients on the Windows 95 and Windows 98 Platform

Server Platforms	**Protocols for Windows 95 and Windows 98 (GUI) Client Platform**
AIX	TCP/IP, DCE
CICS for AIX	CICS Client
HP-UX	TCP/IP
IMS	APPC/IMS
CICS for MVS/ESA	CICS Client
OS/2	TCP/IP, DCE
CICS for OS/2	CICS Client
OS/400	CA/400
Solaris	TCP/IP
CICS for Solaris	CICS Client
VM/ESA	TCP/IP
CICS for VSE/ESA	CICS Client
Windows NT	TCP/IP, DCE
CICS for Windows NT	CICS Client

APPC/IMS Protocol

User Authentication

The user authentication exit provides the user ID and the password specified when the APPC session is allocated. User authentication is described in "User Authentication" on page 21. The program user must be authorized to run the transaction associated with the server call (the transaction program name used on the LU 6.2 connection). For further information, see ″Identifying the Server Location″.

Setting Up Communication Links

The LU 6.2 communication link between the client and server systems must be defined to the host server system and to the client products so that an LU 6.2 session can be allocated between the client and the APPC component of IMS on the host system. Refer to the IMS and communications product documentation for information on setting up the LU 6.2 connection between the client and host systems.

Identifying the Server Location

Use CPIC side information on the client to specify the following information so that the client can allocate the APPC session:
- Partner LU alias
- Transaction program name
- Mode name

The partner LU alias must be the APPC LU name of the IMS Transaction Manager that you want to run the server program. The server program is defined to MVS/APPC.

The following is an example of the command needed to add a transaction program name to APPC/MVS. In this example, tpname must match the transaction program name in the CPIC side profile on the client.

```
TPADD TPSCHED_EXIT(DFSTPPE0)
TPNAME(tpname)
SYSTEM
ACTIVE(YES)
TPSCHED_DELIMITER(##)
TRANCODE=trancode
#
```

Specify the side information identifier in the `location` linkage table attribute for the server program, or set the location dynamically in the client program at run time using EZELOC.

Controlling the Unit of Work

Each server call is an independent unit of work. Any updates made by the server are committed when the server returns to the client. Any EZECOMIT calls issued by the server are ignored. Calling EZEROLLB or a terminating error caught by Server for MVS, VSE, and VM both result in a DL/I ROLB call being issued.

Data Format Conversion

Code format conversion is performed on the server call as specified on the `contable` linkage table attribute and EZECONVT special function word. Code is converted based on the structure of the arguments specified on the CALL statement in the client program.

Error Handling

The standard error handling procedures described in “Communication Error Handling” on page 22 are supported with IMS server calls. EZERT8 is set to the decimal value of the client access service reason code. The reason code is the same as the number portion of error message CSOnnnna as listed in the *VisualAge Generator Messages and Problem Determination Guide* document.

Configuring a Windows 95 and Windows 98 Client for an IMS Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test client programs that call the server program via APPC:

```
►►─:calllink──applname=program name──linktype=──REMOTE────remotecomtype=──APPCIMS────location=──┬─EZELOC──────┬──►
                                                                                                └─system name─┘

►──┬────────────────────┬──┬──────────────────────┬──┬─────────────────────────────────────────┬──────────────►
   └─parmform=──OSLINK──┘  └─luwcontrol=──SERVER──┘  └─contable=──┬─conversion table name─┬─┘
                                                                  ├─'*'───────────────────┤
                                                                  ├─EZECONVT──────────────┤
                                                                  ├─NONE──────────────────┤
                                                                  └─BINARY────────────────┘

►──┬────────────────────────────────┬──────────────────────────────────────────────────────────────────────►◄
   │              ┌─GENERATION─┐    │
   └─remotebind=──┴─RUNTIME────┴────┘
```

The following attributes are ignored:
- externalname
- library
- remoteapptype
- serverid

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=APPCIMS
   location=IMSSIDE parmform=oslink luwcontrol=server contable=ELACNENU
```

Configuring a Windows 95 and Windows 98 Client

Client Access/400 Protocol

User Authentication

If the server program accesses files or relational databases, the user identified on the client must be authorized to run the server program and to access any files or relational tables using dynamic SQL statements.

The program user must start the AS/400 Connection function in the Client Access/400 folder before starting the client program. The connection function prompts for the user ID and password.

Controlling the Unit of Work

Use the `luwcontrol` linkage table attribute to indicate whether server updates are automatically committed on return or whether the client controls the unit of work. All calls to the same OS/400 from the same client session run under the same OS/400 job. If you mix server-controlled unit of work calls with client-controlled unit of work calls, be aware that any commit or rollback, client or server, issued under that job will commit or rollback all outstanding updates in effect for that job.

If you are running Windows GUI clients to call OS/400 servers, a commit or rollback will commit or rollback all outstanding updates for both GUI clients even if you have specified client-controlled unit of work. This is currently due to the behavior of the Client Access/400 product on Windows.

Data Format Conversion

Code format conversion is performed on the call to the server program as directed by the developer using the `contable` linkage table attribute and EZECONVT. Code is converted based on the structure of the parameters specified on the server call in the client program.

Error Handling

The standard error handling procedures described in “Communication Error Handling” on page 22 are supported with Client Access/400. EZERT8 is set to the decimal value of the client access service reason code. The reason code is the same as the number portion of error message CSOnnnna as listed in the *VisualAge Generator Messages and Problem Determination Guide* document.

Any error messages logged on the server are spooled to the OS/400 user ID of the user signed on to Client Access/400 on the client system. The job log will contain all messages logged since the job was started. The client access job is a prestarted job used repeatedly by many users; therefore, the job log may contain messages from other remote commands. Move to the bottom of the job log to see the last set of error messages.

Configuring a Windows 95 and Windows 98 Client for an OS/400 Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call OS/400 server programs via CA/400:

```
►►—:calllink—applname=program name—contable=—┬—conversion table name—┬——————————►
                                             ├—'*'———————————————————┤
                                             ├—EZECONVT——————————————┤
                                             ├—NONE——————————————————┤
                                             └—BINARY————————————————┘

►—linktype=—REMOTE—library=library name—remotecomtype=—CA400———————————————————►

►—┬——————————————————┬—┬————————————————————————————┬————————————————————————————►
  └—parmform=—OSLINK—┘ │                ┌—applname—┐ │
                       └—externalname=——┴——————————┴—┘

►—┬————————————————————————————┬—┬————————————————————————————┬———————————————————►
  └—location=—┬—EZELOC——————┬——┘ │              ┌—CLIENT—┐    │
              └—system name—┘    └—luwcontrol=——┴—SERVER—┴————┘

►—┬————————————————————————————┬—┬——————————————————————————————————┬——————————————►◄
  └—remoteapptype=—┬—VG————┬———┘ │              ┌—GENERATION—┐      │
                   ├—NONVG—┤     └—remotebind=——┴—RUNTIME————┴——————┘
                   └—ITF———┘
```

The following attributes are ignored:
- serverid

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CA400
contable=ELACNENU library=ELACVP5 parmform=OSLINK location=SILVER6
```

Client Access/400 Set Up: Client programs use the Client Access/400 Remote Command/Distributed Program Call APIs to call the generated server program. Ensure that you have done the following before attempting to run the VisualAge Generator client program:

- Install the appropriate Client Access/400 product on the Windows client system
- Establish the communication link between Client Access/400 and the OS/400 server system. Client Access/400 will then attempt to start the communication link when the VisualAge Generator program issues the remote command.

Configuring a Windows 95 and Windows 98 Client

Your Client Access/400 set up determines whether LU 6.2 or TCP/IP is used for communication.

For further information, refer to the following manual:

- *Client Access/400 Optimized for OS/2 - Getting Started*, SC41-3510.

CICS Client Protocol

User Authentication

The user authentication exit (see "User Authentication" on page 21) provides the user ID and the password specified on the ECI call. The program user must be authorized to run the transaction associated with the server call.

The user exit can return NULLs for the userid and password. The default exit returns the contents of the environment variables CSOUID and CSOPWD as the userid and password. If nulls are specified on the ECI call, the CICS Client determines the user ID and password in a system-dependent fashion. Refer to the CICS Client documentation for your environment for further information.

Setting Up Communication Links

The communication link between the client and server systems must be defined to the CICS server system and client products to allow an ECI call to flow from a CICS Client product to server systems. CICS Client, the communications software, is installed and configured on the client. Refer to CICS CLIENT documentation. The CICS server environment must have a "listener" defined and requires other entries if remote programs are called. Refer to CICS documentation for more information. Refer to the CICS intercommunication documentation for your CICS systems and client products for additional information.

Identifying the CICS Transaction for the Server

The CICS transaction name associated with the server program is specified in the `serverid` linkage table attribute. If not specified, the default transaction is the CICS-supplied mirror transaction, CPMI.

Controlling the Unit of Work

Extended Units of Work: Multiple synchronous calls to CICS servers can be issued from the same client. You can use the extended unit of work feature of ECI to include several calls to the same system within the same unit of work by specifying `luwcontrol=CLIENT` in the linkage table for the server programs. For CICS servers, the default value for LUWCONTROL is client unit of work. The extended unit of work ends when the client program calls EZECOMIT or EZEROLLB, which results in an ECI call to commit or roll back any extended transactions that are currently active.

A separate ECI extended unit of work (CICS transaction) is started for each unique `serverid/location` pair. Servers on the same system running under the same SERVERID (transaction name) are part of the same CICS extended transaction. A client EZECOMIT or ROLLBACK call ends all the extended transactions currently in effect for the client.

The server cannot issue EZECOMIT or EZEROLLB calls if the client unit of work was in effect for the server call.

Server Unit of Work: You can specify `luwcontrol=SERVER` in the linkage table. With this specification, each server call is a separate unit of work. The server program can issue commit or rollback requests as well.

Data Format Conversion

Code format conversion is performed on the server call as specified on the `contable` linkage table attribute and EZECONVT special function word. Code is converted based on the structure of the arguments specified on the CALL statement in the client program.

Error Handling

The standard error handling procedures described in "Communication Error Handling" on page 22 are supported with CICS ECI. EZERT8 is set to the decimal value of the client access service reason code. The reason code is the same as the number portion of error message CSOnnnna as listed in the *VisualAge Generator Messages and Problem Determination Guide* document.

Specifying Parameter Format

Parameter format (the `parmform` attribute on the calllink tag) can be specified as COMMDATA (data values passed in the COMMAREA) or COMMPTR (pointers passed in the COMMAREA).

Using COMMPTR Format: Use COMMPTR for calling COBOL subroutines that were developed to use the non-VAGen parameter passing convention that uses pointers in the CICS COMMAREA. When the pointer format is used, the generated client program calls an intermediate catcher program (ELACSV, provided with VisualAge Generator host services) via the ECI instead of calling the server program directly. The catcher program receives the parameter data from the client, builds a pointer list in the COMMAREA that points to the individual parameters, and links to the server program. A program processing table (PPT) must be defined for the ELACSV program on the CICS server system.

COMMPTR format is only supported for MVS CICS, CICS for VSE/ESA, and CICS OS/2 servers.

Using COMMDATA Format: Use COMMDATA for calling generated programs and for calling COBOL subroutines where possible. COMMDATA provides better performance in calling remote programs. With COMMDATA, the server program is called directly without going through the catcher program.

Configuring a Windows 95 and Windows 98 Client for a CICS for AIX Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─CICSCLIENT──────►

►──┬──────────────────────────────────────┬──┬───────────────────────────┬────────────►
   └─contable=──┬─conversion table name─┬─┘  └─location=──┬─EZELOC──────┬─┘
                ├─'*'───────────────────┤                 └─system name─┘
                ├─EZECONVT──────────────┤
                ├─NONE──────────────────┤
                └─BINARY────────────────┘

►──┬─────────────────────────┬──┬──────────────────────────┬──────────────────────────►
   │             ┌─CLIENT─┐  │  └─parmform=──┬─OSLINK────┬──┘
   └─luwcontrol=─┴─SERVER─┴──┘               ├─COMMDATA──┤
                                             └─CICSOSLINK┘

►──┬─────────────────────────────┬──┬──────────────────────────────┬──────────────────►
   └─remoteapptype=──┬─VG────┬───┘  │              ┌─GENERATION─┐  │
                     ├─NONVG─┤      └─remotebind=──┴─RUNTIME────┴──┘
                     └─ITF───┘

►──┬──────────────────────────────────┬───────────────────────────────────────────────►◄
   └─serverid=──server identifier─────┘
```

The following attributes are ignored:
- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST contable=BINARY
```

Configuring a Windows 95 and Windows 98 Client for a CICS for OS/2 Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─CICSCLIENT────────►

►─┬────────────────────────────────────────┬─┬───────────────────────────────┬────────►
  └─contable=─┬─conversion table name─┬─┘ └─location=─┬─EZELOC──────┬─┘
              ├─'*' ─────────────────┤                  └─system name─┘
              ├─EZECONVT─────────────┤
              ├─NONE─────────────────┤
              └─BINARY───────────────┘

►─┬─────────────────────────────┬─┬─────────────────────────┬─────────────────────────►
  │               ┌─CLIENT─┐    │ └─parmform=─┬─OSLINK─────┬─┘
  └─luwcontrol=──┴─SERVER─┴─┘             ├─COMMPTR────┤
                                           ├─COMMDATA───┤
                                           └─CICSOSLINK─┘

►─┬──────────────────────────────┬─┬──────────────────────────────────┬────────────────►
  └─remoteapptype=─┬─VG─────┬─┘ │                ┌─GENERATION─┐   │
                   ├─NONVG──┤     └─remotebind=──┴─RUNTIME────┴─┘
                   └─ITF────┘

►─┬───────────────────────────────────┬───────────────────────────────────────────►◄
  └─serverid=──server identifier─┘
```

The following attributes are ignored:
- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST
```

Configuring a Windows 95 and Windows 98 Client for a CICS for Windows NT Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─CICSCLIENT────────►
```

```
►─┬─────────────────────────────────────┬──┬──────────────────────────────┬──────────►
  └─contable=──┬─conversion table name─┬─┘  └─location=──┬─EZELOC──────┬─┘
               ├─'*' ──────────────────┤                 └─system name─┘
               ├─EZECONVT──────────────┤
               ├─NONE──────────────────┤
               └─BINARY────────────────┘

►─┬─────────────────────────┬──┬─────────────────────────────┬────────────────────────►
  │            ┌─CLIENT─┐   │  └─parmform=──┬─OSLINK────┬──┘
  └─luwcontrol=┴─SERVER─┴───┘               ├─COMMDATA──┤
                                            └─CICSOSLINK┘

►─┬─────────────────────────────┬──┬─────────────────────────────────┬────────────────►
  └─remoteapptype=──┬─VG────┬───┘  │              ┌─GENERATION─┐     │
                    ├─NONVG─┤      └─remotebind=──┴─RUNTIME────┴─────┘
                    └─ITF───┘

►─┬─────────────────────────────────────┬─────────────────────────────────────────────►◄
  └─serverid=──server identifier────────┘
```

The following attributes are ignored:
- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST
```

Configuring a Windows 95 and Windows 98 Client for an MVS CICS Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─CICSCLIENT──────►

►─┬─────────────────────────────────────┬──┬──────────────────────────────┬──────────►
  └─contable=──┬─conversion table name─┬─┘  └─location=──┬─EZELOC──────┬─┘
               ├─'*' ──────────────────┤                 └─system name─┘
               ├─EZECONVT──────────────┤
               ├─NONE──────────────────┤
               └─BINARY────────────────┘
```

```
►─┬──────────────────────────┬─┬──────────────────────────┬──────────────────►
  │               ┌─CLIENT─┐ │ └─parmform=─┬─OSLINK─────┬─┘
  └─luwcontrol=───┴─SERVER─┴─┘             ├─COMMPTR────┤
                                           ├─COMMDATA───┤
                                           └─CICSOSLINK─┘

►─┬────────────────────────────┬─┬──────────────────────────────┬────────────►
  └─remoteapptype=─┬─VG────────┘ │               ┌─GENERATION─┐ │
                   ├─NONVG─┤     └─remotebind=───┴─RUNTIME────┴─┘
                   └─ITF───┘

►─┬─────────────────────────────────┬────────────────────────────────────────►◄
  └─serverid=──server identifier────┘
```

The following attributes are ignored:
- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST contable=ELACNENU
```

Configuring a Windows 95 and Windows 98 Client for a VSECICS Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─CICSCLIENT──────►

►─┬──────────────────────────────────────┬─┬──────────────────────────────┬──────►
  └─contable=─┬─conversion table name──┬─┘ └─location=─┬─EZELOC──────┬─┘
              ├─'*'────────────────────┤               └─system name─┘
              ├─EZECONVT───────────────┤
              ├─NONE───────────────────┤
              └─BINARY─────────────────┘

►─┬──────────────────────────┬─┬──────────────────────────┬──────────────────►
  │               ┌─CLIENT─┐ │ └─parmform=─┬─OSLINK─────┬─┘
  └─luwcontrol=───┴─SERVER─┴─┘             ├─COMMPTR────┤
                                           ├─COMMDATA───┤
                                           └─CICSOSLINK─┘
```

```
►──┬─────────────────────────────┬──┬────────────────────────────────┬─────────►
   └─remoteapptype=──┬─VG────┬───┘  └─remotebind=──┬─GENERATION──┬──┘
                     ├─NONVG─┤                     └─RUNTIME─────┘
                     └─ITF───┘

►──┬─────────────────────────────────┬───────────────────────────────────────►◄
   └─serverid=───server identifier───┘
```

The following attributes are ignored:
- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST contable=ELACNENU
```

Configuring a Windows 95 and Windows 98 Client for a CICS for Solaris Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─CICSCLIENT──────►

►──┬─────────────────────────────────────────┬──┬─────────────────────────────┬─────────►
   └─contable=──┬─conversion table name──┬───┘  └─location=──┬─EZELOC──────┬──┘
                ├─'*'────────────────────┤                   └─system name─┘
                ├─EZECONVT───────────────┤
                ├─NONE───────────────────┤
                └─BINARY─────────────────┘

►──┬──────────────────────────────┬──┬──────────────────────────────┬───────────────────►
   └─luwcontrol=──┬─CLIENT─┬──────┘  └─parmform=──┬─OSLINK─────┬────┘
                  └─SERVER─┘                      ├─COMMDATA───┤
                                                  └─CICSOSLINK─┘

►──┬─────────────────────────────┬──┬────────────────────────────────┬─────────►
   └─remoteapptype=──┬─VG────┬───┘  └─remotebind=──┬─GENERATION──┬──┘
                     ├─NONVG─┤                     └─RUNTIME─────┘
                     └─ITF───┘

►──┬─────────────────────────────────┬───────────────────────────────────────►◄
   └─serverid=───server identifier───┘
```

The following attributes are ignored:

- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST
```

DCE Protocol

Processing Flow for VisualAge Generator DCE Common Services Remote Call

This section shows the processing flow for a VisualAge Generator DCE common services remote call.

1. VisualAge Generator DCE Server is started using a configuration file which specifies the DCE principal name that the server will obtain its DCE authorizations from (equivalent to a DCE userid), the location and the serverid name for binding information advertising, the Access Control List (ACL) object for client authorization, and the server programs that the server is authorized to process. The server programs are specified in one of two different groups; those in which secure DCE (authenticated RPC) communications are required and those which can be accessed via unsecured (unauthenticated RPC) DCE communications.

   The server obtains an object UUID for each program from the CDS object /.:/Servers/VAGenerator/SERVERID/program. If the program object is not defined, one will be created for it. The server uses the program object UUIDs when it advertises its binding information.
2. VisualAge Generator DCE Server advertises each of the programs that it services. The Cell Directory Services (CDS) location used for advertising the binding information is /.:/Servers/VAGenerator/SERVERID/LOCATION.
3. VisualAge Generator client retrieves the object UUID for the program from /.:/Servers/VAGenerator/SERVERID/program-name. It then uses the object UUID to request the binding information for a server which services the program from the CDS location /.:/Servers/VAGenerator/SERVERID/LOCATION. If there are multiple server bindings that match the search criteria, DCE will randomly return one of them.
4. VisualAge Generator client will setup for authenticated RPC, if DCESECURE is specified in the linkage table.
5. VisualAge Generator client performs data conversion on passed parameters as specified in the `contable` attribute of the client linkage table (runtime or generation time as applicable).
6. VisualAge Generator client makes remote call to VisualAge Generator DCE Server.

7. VisualAge Generator DCE Server checks if VisualAge Generator client is authorized to use server program (via DCE CDS Access Control List) and if the appropriate level of communication security is being used from the client.
8. VisualAge Generator DCE Server checks if the server program requested is one that it is authorized to process (via initial configuration file).
9. VisualAge Generator DCE Server processes client request, closes Logical Unit of Work, and returns data to client.

User Authentication and Authorization

User authentication is performed on the client via the DCE security server using the client's DCE login identifier. The VisualAge Generator DCE server checks whether the client is authorized to call the DCE server using DCE ACL security services. User authorization is performed via the DCE ACL security services. The VisualAge Generator DCE server is told at startup time the DCE object ACL to use for checking client authorization for running the called server programs. There is only one ACL used per VisualAge Generator DCE server; therefore, the authorization is at the VisualAge Generator DCE server level and not the called program level. If a called program requires a special ACL, then another VisualAge Generator DCE server will have to be created or started. The test ACL attribute determines whether or not the client is authorized to execute the server program (if the client has test privileges on the ACL object, then the client is authorized to execute all server programs provided by the server).

Controlling the Unit of Work

All calls via the DCE common services are server units of work. All resource changes are committed when the server returns to the calling program.

Data Format Conversion

Code format conversion is performed on the server call as specified on the `contable` attribute for the server entry in the linkage table. Data is converted based on the structure of the parameters specified on the server call in the client program.

Error Handling

The standard error handling procedures described in "Communication Error Handling" on page 22 are supported with VisualAge Generator DCE common services. EZERT8 is set to the decimal value of the client access service reason code. EZERT8 is only set when the REPLY option is coded on the call to the remote server program. If a visual link is used to make the call from a GUI program, the REPLY option is used on the call.

Any errors trapped by DCE are passed to the client program with a corresponding CSO error message. The error message contains an insert with the DCE mnemonic. A non-zero return code from the called program is passed back to the client program with a corresponding CSO error message. The error message contains an insert with the return code from the called program. VisualAge Generator return codes are documented in the help for the message.

All errors are traced to the CSO trace file on the client and server machines, as applicable.

Configuring a Windows 95 and Windows 98 Client for an OS/2 Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call server programs via the DCE common services:

```
►►──:calllink──applname=program name──linktype=─REMOTE──────────────────────────────►

►─remotecomtype=──┬─DCE───────┬──┬──────────────────────┬───────────────────────────►
                  └─DCESECURE─┘  └─parmform=─COMMDATA───┘

►──┬──────────────────────────────────────────┬──┬──────────────────────────────┬───►
   └─contable=──┬─conversion table name──┬────┘  └─location=──┬─EZELOC──────┬───┘
                ├─'*' ───────────────────┤                    └─system name─┘
                ├─EZECONVT───────────────┤
                ├─NONE───────────────────┤
                └─BINARY─────────────────┘

►──┬─────────────────────────────────────┬──┬───────────────────────────────┬──────►◄
   │                ┌─GENERATION─┐       │  └─serverid=───server identifier─┘
   └─remotebind=────┴─RUNTIME────┴───────┘
```

The following attributes are ignored:
- remoteapptype
- externalname
- library
- luwcontrol

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=DCE
   location=Server1 remotebind=RUNTIME serverid=Test
```

Configuring a Windows 95 and Windows 98 Client

Configuring a Windows 95 and Windows 98 Client for a Windows NT Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call server programs via the DCE common services:

```
►►──:calllink──applname=program name──linktype=─REMOTE──────────────────────────►

►─remotecomtype=──┬─DCE───────┬──┬────────────────────┬─────────────────────────►
                  └─DCESECURE─┘  └─parmform=─COMMDATA─┘

►──┬──────────────────────────────────────┬──┬──────────────────────────────┬───►
   └─contable=──┬─conversion table name─┬─┘  └─location=──┬─EZELOC──────┬──┘
                ├─'*'───────────────────┤                 └─system name─┘
                ├─EZECONVT──────────────┤
                ├─NONE──────────────────┤
                └─BINARY────────────────┘

►──┬────────────────────────────────┬──┬──────────────────────────────────┬─────►◄
   │              ┌─GENERATION─┐    │  └─serverid=──server identifier─────┘
   └─remotebind=──┴─RUNTIME────┴────┘
```

The following attributes are ignored:
- remoteapptype
- externalname
- library
- luwcontrol

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=DCE
   location=Server1 remotebind=RUNTIME serverid=Test
```

Configuring a Windows 95 and Windows 98 Client for an AIX Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call server programs via the DCE common services:

```
►►──:calllink──applname=program name──linktype=─REMOTE──────────────────────────►
```

```
►─remotecomtype=─┬─DCE───────┬──┬────────────────────┬──────────────────────────►
                 └─DCESECURE─┘  └─parmform=─COMMDATA─┘

►─┬──────────────────────────────────────┬──┬──────────────────────────────┬────►
  └─contable=─┬─conversion table name─┬──┘  └─location=─┬─EZELOC──────┬────┘
              ├─'*'───────────────────┤                 └─system name─┘
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬─────────────────────────────────┬──┬──────────────────────────────────┬─────►◄
  │                ┌─GENERATION─┐   │  └─serverid=───server identifier────┘
  └─remotebind=────┴─RUNTIME────┴───┘
```

The following attributes are ignored:
- remoteapptype
- externalname
- library
- luwcontrol

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=DCE
   location=Server1 contable=BINARY remotebind=RUNTIME serverid=Test
```

TCP/IP Protocol

Creating a TCP/IP Services File Entry

The client machine must have an entry for the TCP/IP Serverid added to its TCP/IP services file. For Windows, AIX, HP-UX, and Solaris the TCP/IP SERVICES file is used.

Error Handling

The standard error handling procedures described in "Communication Error Handling" on page 22 are supported with TCPIP server calls. EZERT8 is set to the decimal value of the client access service reason code. The reason code is the same as the number portion of error message CSOnnnna as listed in the *VisualAge Generator Messages and Problem Determination Guide* document.

Configuring an Windows 95 and Windows 98 Client for an OS/2 Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─TCPIP──────────►
```

```
►─┬──────────────────────────────────────┬──────────────────────────────────────►
  └─contable=─┬─conversion table name─┬─┘
              ├─'*' ──────────────────┤
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬─────────────────────────────────────────┬───────────────────────────────────►
  └─location=──system name_tcpip_hostname───┘

►─┬─────────────────────────────────────┬──────────────────────────────────────►◄
  └─serverid=──tcpip_server identifier──┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to "Conversion Table by Language and Platform" on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 remotebind=RUNTIME serverid=port1
```

Configuring an Windows 95 and Windows 98 Client for a Windows NTServer

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►─:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─TCPIP──────►

►─┬──────────────────────────────────────┬──────────────────────────────────────►
  └─contable=─┬─conversion table name─┬─┘
              ├─'*' ──────────────────┤
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬─────────────────────────────────────────┬───────────────────────────────────►
  └─location=──system name_tcpip_hostname───┘
```

```
►─┬──────────────────────────────────┬────────────────────────────────────────►◄
  └─serverid=──tcpip_server identifier─┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to "Conversion Table by Language and Platform" on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 remotebind=RUNTIME serverid=port1
```

Configuring an Windows 95 and Windows 98 Client for an AIX Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─TCPIP──────────►

►─┬─────────────────────────────────────┬──────────────────────────────────────────►
  └─contable=─┬─conversion table name─┬─┘
              ├─'*' ──────────────────┤
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬──────────────────────────────────────────┬─────────────────────────────────────►
  └─location=──system name_tcpip_hostname────┘

►─┬──────────────────────────────────┬────────────────────────────────────────►◄
  └─serverid=──tcpip_server identifier─┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to "Conversion Table by Language and Platform" on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 contable=BINARY remotebind=RUNTIME serverid=port1
```

Configuring an Windows 95 and Windows 98 Client for an HP-UX Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─TCPIP──────────►

►──┬──────────────────────────────────┬──────────────────────────────────────────────►
   └─contable=─┬─conversion table name─┬─┘
               ├─'*' ──────────────────┤
               ├─EZECONVT──────────────┤
               ├─NONE──────────────────┤
               └─BINARY────────────────┘

►──┬───────────────────────────────────────┬─────────────────────────────────────────►
   └─location=───system name_tcpip_hostname───┘

►──┬────────────────────────────────────┬───────────────────────────────────────────►◄
   └─serverid=───tcpip_server identifier───┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to "Conversion Table by Language and Platform" on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 contable=BINARY remotebind=RUNTIME serverid=port1
```

Configuring an Windows 95 and Windows 98 Client for a Solaris Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─TCPIP──────────►
```

```
►─┬──────────────────────────────────┬──────────────────────────────────────────────►
  └─contable=─┬─conversion table name─┬─┘
              ├─'*' ──────────────────┤
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬─────────────────────────────────────────────┬───────────────────────────────────►
  └─location=───system name_tcpip_hostname──────┘

►─┬──────────────────────────────────────────┬───────────────────────────────────►◄
  └─serverid=───tcpip_server identifier──────┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to "Conversion Table by Language and Platform" on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 contable=BINARY remotebind=RUNTIME serverid=port1
```

Configuring an Windows 95 and Windows 98 Client for a VM/ESA Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─TCPIP────────►

►─┬──────────────────────────────────┬──────────────────────────────────────────────►
  └─contable=─┬─conversion table name─┬─┘
              ├─'*' ──────────────────┤
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬─────────────────────────────────────────────┬───────────────────────────────────►
  └─location=───system name_tcpip_hostname──────┘
```

Configuring a Windows 95 and Windows 98 Client

```
►─┬──────────────────────────────────┬──────────────────────────────────────────►◄
  └─serverid=──tcpip_server identifier──┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to "Conversion Table by Language and Platform" on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 contable=ELACNENU remotebind=RUNTIME serverid=port1
```

Configuring a Windows 95 and Windows 98 Server

Summary Table of Valid Clients and Protocols

Table 30. Valid Clients and Protocols for VisualAge Generator Servers on the Windows 95 and Windows 98 Platform

Server Platforms	**Client Platforms**					
	OS/2 (GUI, C++, ITF)	**Windows 95 and Windows 98 (GUI)**	**Windows NT (GUI, C++, ITF)**	**AIX (C++)**	**CICS**	**Solaris (C++)**
Windows 95 and Windows 98	N/A	N/A	N/A	N/A	N/A	N/A

Notes:
N/A = Not available
CICS as a Client Platform refers to:
- CICS for AIX
- CICS for MVS/ESA
- CICS for OS/2
- CICS for Windows NT
- CICS for VSE/ESA
- CICS for Solaris

Chapter 16. Windows NT Platform

How to use this chapter

This chapter contains platform and protocol information necessary to configure a client program or a server program. The chapter is divided into configuration sections: client programs and server programs. Each configuration section is organized by protocol. Each protocol section contains information organized by target platforms.

Use the following method to locate the information you need:

1. Verify that this chapter is appropriate to your runtime environment.

   Ensure that the runtime environment of your program matches the platform on which this chapter is based. Turn to the appropriate chapter if the runtime environment of your program does not match the platform named in the title of this chapter.
2. Determine which type of program you want to configure: client program or server program.
3. Determine which configuration section in the chapter is appropriate for your program.
4. Find the intended target platform of the program you are configuring in the Summary Table of Valid Clients and Protocols or Summary Table of Valid Servers and Protocols.

   If you are configuring a client program, find the target platform of the intended server program in the column of valid platforms.

   If you are configuring a server program, find the target platform of the intended client program in the row of valid platforms.
5. Chose a protocol based on the information in the table.

   If you are configuring a client program, find the cell in the table where the row of the intended target platform intersects the column of the client platform. Select a protocol from the list of available protocols for this client/server combination.

   If you are configuring a client program, find the cell in the table where the column of the intended target platform intersects the row of the server platform. Select a protocol from the list of available protocols for this client/server combination.
6. Determine the name of the appropriate protocol section within the configuration section.

Turn to the part of the configuration section indicated by the protocol name. Here you will find information common to all target platforms valid for this protocol.

7. Determine the name of the appropriate target platform section within the protocol section.

   You will find platform specific information for the chosen protocol in the section named for your intended target platform.

Example

```
Chapter: Windows NT Platform
Type of program to configure: Client program
Configuration section: Configuring a Windows NT client
Intended target platform: CICS for MVS/ESA (i.e. a CICS for MVS/ESA server program)
Chosen protocol: CICS Client
Protocol section: CICS Client Protocol
Target platform section: Configuring a Windows NT Client for a
CICS for MVS/ESA Server
```

You will encounter syntax diagrams in this chapter. The diagrams are used to show programming syntax as follows:

```
►►──:RequiredTerm──RequiredTerm=RequiredValue──RequiredTerm=──┬─OptionalValue─┬──────────►
                                                              └─OptionalValue─┘

►──┬──────────────────────────────────────────┬──┬─────────────────────────────────┬──────►
   │                  ┌─DefaultValue──┐       │  └─OptionalTerm=RequiredValue──────┘
   └─OptionalTerm=────┴─OptionalValue─┴───────┘

►──┬──────────────────────────────────────────┬────────────────────────────────────────►◄
   └─OptionalTerm=────┬─OptionalValue─┬───────┘
                      └─OptionalValue─┘
```

RequiredTerm	A required term is a term that must be specified. Required terms are depicted in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required terms.
RequiredValue	A required value is a value that must be specified for a term. A required value is shown in italics in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required values.
OptionalTerm	An optional term is a term that can be

	specified. An optional term is depicted in syntax diagrams in an alternate path from the preceding term.
OptionalValue	An optional value is a value that can be specified for a term. An optional value is shown in italics in syntax diagrams in an alternate path from the preceding term along with other optional values or a default value.
DefaultValue	A default value is the value that is specified by default for a term. A default value is shown in italics in syntax diagrams in an alternate path above other optional values.

Configuring a Windows NT Client

Summary Table of Valid Servers and Protocols

Table 31. Valid Servers and Protocols for VisualAge Generator Clients on the Windows NT Platform

Server Platforms	Protocols for Windows NT (GUI, C++, ITF) Client Platform
AIX	TCP/IP, DCE
CICS for AIX	CICS Client
HP-UX	TCP/IP
IMS	APPC/IMS
CICS for MVS/ESA	CICS Client
OS/2	TCP/IP, DCE
CICS for OS/2	CICS Client
OS/400	CA/400
Solaris	TCP/IP
CICS for Solaris	CICS Client
VM/ESA	TCP/IP
CICS for VSE/ESA	CICS Client
Windows NT (C++)	TCP/IP, DCE, IPC, DIRECT
Windows NT (Java)	TCP/IP, DIRECT
CICS for Windows NT	CICS Client

APPC/IMS Protocol

User Authentication

The user authentication exit provides the user ID and the password specified when the APPC session is allocated. User authentication is described in “User Authentication” on page 21. The program user must be authorized to run the transaction associated with the server call (the transaction program name used on the LU 6.2 connection). For further information, see ″Identifying the Server Location″.

Setting Up Communication Links

The LU 6.2 communication link between the client and server systems must be defined to the host server system and to the client products so that an LU 6.2 session can be allocated between the client and the APPC component of IMS on the host system. Refer to the IMS and communications product documentation for information on setting up the LU 6.2 connection between the client and host systems.

Identifying the Server Location

Use CPIC side information on the client to specify the following information so that the client can allocate the APPC session:
- Partner LU alias
- Transaction program name
- Mode name

The partner LU alias must be the APPC LU name of the IMS Transaction Manager that you want to run the server program. The server program is defined to MVS/APPC.

The following is an example of the command needed to add a transaction program name to APPC/MVS. In this example, tpname must match the transaction program name in the CPIC side profile on the client.

```
TPADD TPSCHED_EXIT(DFSTPPE0)
TPNAME(tpname)
SYSTEM
ACTIVE(YES)
TPSCHED_DELIMITER(##)
TRANCODE=trancode
#
```

Specify the side information identifier in the `location` linkage table attribute for the server program, or set the location dynamically in the client program at run time using EZELOC.

Controlling the Unit of Work

Each server call is an independent unit of work. Any updates made by the server are committed when the server returns to the client. Any EZECOMIT calls issued by the server are ignored. Calling EZEROLLB or a terminating error caught by Server for MVS, VSE, and VM both result in a DL/I ROLB call being issued.

Data Format Conversion

Code format conversion is performed on the server call as specified on the `contable` linkage table attribute and EZECONVT special function word. Code is converted based on the structure of the arguments specified on the CALL statement in the client program.

Error Handling

The standard error handling procedures described in “Communication Error Handling” on page 22 are supported with IMS server calls. EZERT8 is set to the decimal value of the client access service reason code. The reason code is the same as the number portion of error message CSOnnnna as listed in the *VisualAge Generator Messages and Problem Determination Guide* document.

Configuring a Windows NT Client for an IMS Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test client programs that call the server program via APPC:

```
►►─:calllink──applname=program name───linktype=───REMOTE─────remotecomtype=───APPCIMS─────location=──┬─EZELOC──────┬──►
                                                                                                   └─system name─┘

►─┬────────────────────┬─┬─────────────────────┬─┬──────────────────────────────────────┬──────────────────────────────►
  └─parmform=───OSLINK─┘ └─luwcontrol=───SERVER─┘ └─contable=──┬─conversion table name─┬─┘
                                                              ├─'*'──────────────────┤
                                                              ├─EZECONVT─────────────┤
                                                              ├─NONE─────────────────┤
                                                              └─BINARY───────────────┘

►─┬─────────────────────────────────┬─────────────────────────────────────────────────────────────────────────────►◄
  │                ┌─GENERATION─┐   │
  └─remotebind=───┴─RUNTIME────┴───┘
```

The following attributes are ignored:
- externalname
- library
- remoteapptype
- serverid

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=APPCIMS
   location=IMSSIDE parmform=oslink luwcontrol=server contable=ELACNENU
```

Client Access/400 Protocol

This chapter contains information specific to implementing client/server calls using Client Access/400 (CA/400). Be sure to read the general information in "Chapter 2. Introduction to Client/Server Processing with Synchronous Calls" on page 9 before reading this chapter.

User Authentication

If the server program accesses files or relational databases, the user identified on the client must be authorized to run the server program and to access any files or relational tables using dynamic SQL statements.

The program user must start the AS/400 Connection function in the Client Access/400 folder before starting the client program. The connection function prompts for the user ID and password.

Controlling the Unit of Work

Use the `luwcontrol` linkage table attribute to indicate whether server updates are automatically committed on return or whether the client controls the unit of work. All calls to the same OS/400 from the same client session run under the same OS/400 job. If you mix server-controlled unit of work calls with client-controlled unit of work calls, be aware that any commit or rollback, client or server, issued under that job will commit or rollback all outstanding updates in effect for that job.

If you are running Windows GUI clients to call OS/400 servers, a commit or rollback will commit or rollback all outstanding updates for both GUI clients even if you have specified client-controlled unit of work. This is currently due to the behavior of the Client Access/400 product on Windows.

Data Format Conversion

Code format conversion is performed on the call to the server program as directed by the developer using the `contable` linkage table attribute and EZECONVT. Code is converted based on the structure of the parameters specified on the server call in the client program.

Error Handling

The standard error handling procedures described in "Communication Error Handling" on page 22 are supported with Client Access/400. EZERT8 is set to the decimal value of the client access service reason code. The reason code is the same as the number portion of error message CSOnnnna as listed in the *VisualAge Generator Messages and Problem Determination Guide* document.

Any error messages logged on the server are spooled to the OS/400 user ID of the user signed on to Client Access/400 on the client system. The job log will contain all messages logged since the job was started. The client access job is a prestarted job used repeatedly by many users; therefore, the job log may contain messages from other remote commands. Move to the bottom of the job log to see the last set of error messages.

Configuring a Windows NT Client for an OS/400 Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call OS/400 server programs via CA/400:

```
►►──:calllink──applname=program name──contable=──┬─conversion table name─┬──────────►
                                                 ├─'*' ──────────────────┤
                                                 ├─EZECONVT──────────────┤
                                                 ├─NONE──────────────────┤
                                                 └─BINARY────────────────┘

►─linktype=─REMOTE──library=library name──remotecomtype=─CA400────────────────────►

►──┬─────────────────────┬──┬────────────────────────────────┬────────────────────►
   └─parmform=─OSLINK────┘  │                ┌─applname─┐    │
                            └─externalname=──┴──────────┴────┘

►──┬─────────────────────────────────┬──┬─────────────────────────────┬───────────►
   └─location=──┬─EZELOC──────────┬──┘  │              ┌─CLIENT─┐     │
                └─system name─────┘     └─luwcontrol=──┴─SERVER─┴─────┘

►──┬────────────────────────────┬──┬─────────────────────────────────┬────────────►◄
   └─remoteapptype=──┬─VG────┬──┘  │              ┌─GENERATION─┐     │
                     ├─NONVG─┤     └─remotebind=──┴─RUNTIME────┴─────┘
                     └─ITF───┘
```

The following attributes are ignored:
- serverid

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CA400
contable=ELACNENU library=ELACVP5 parmform=OSLINK location=SILVER6
```

Client Access/400 Set Up: Client programs use the Client Access/400 Remote Command/Distributed Program Call APIs to call the generated server program. Ensure that you have done the following before attempting to run the VisualAge Generator client program:

- Install the appropriate Client Access/400 product on the Windows client system

- Establish the communication link between Client Access/400 and the OS/400 server system. Client Access/400 will then attempt to start the communication link when the VisualAge Generator program issues the remote command.

Your Client Access/400 set up determines whether LU 6.2 or TCP/IP is used for communication.

For further information, refer to the following manual:

- *Client Access/400 Optimized for OS/2 - Getting Started*, SC41-3510.

CICS Client Protocol

User Authentication

The user authentication exit (see "User Authentication" on page 21) provides the user ID and the password specified on the ECI call. The program user must be authorized to run the transaction associated with the server call.

The user exit can return NULLs for the userid and password. The default exit returns the contents of the environment variables CSOUID and CSOPWD as the userid and password. If nulls are specified on the ECI call, the CICS Client determines the user ID and password in a system-dependent fashion. Refer to the CICS Client documentation for your environment for further information.

Setting Up Communication Links

The communication link between the client and server systems must be defined to the CICS server system and client products to allow an ECI call to flow from a CICS Client product to server systems. CICS Client, the communications software, is installed and configured on the client. Refer to CICS CLIENT documentation. The CICS server environment must have a "listener" defined and requires other entries if remote programs are called. Refer to CICS documentation for more information. Refer to the CICS intercommunication documentation for your CICS systems and client products for additional information.

Identifying the CICS Transaction for the Server

The CICS transaction name associated with the server program is specified in the `serverid` linkage table attribute. If not specified, the default transaction is the CICS-supplied mirror transaction, CPMI.

Controlling the Unit of Work

Extended Units of Work: Multiple synchronous calls to CICS servers can be issued from the same client. You can use the extended unit of work feature of ECI to include several calls to the same system within the same unit of work by specifying `luwcontrol=CLIENT` in the linkage table for the server programs. For CICS servers, the default value for LUWCONTROL is client unit of work.

The extended unit of work ends when the client program calls EZECOMIT or EZEROLLB, which results in an ECI call to commit or roll back any extended transactions that are currently active.

A separate ECI extended unit of work (CICS transaction) is started for each unique `serverid/location` pair. Servers on the same system running under the same SERVERID (transaction name) are part of the same CICS extended transaction. A client EZECOMIT or ROLLBACK call ends all the extended transactions currently in effect for the client.

The server cannot issue EZECOMIT or EZEROLLB calls if the client unit of work was in effect for the server call.

Server Unit of Work: You can specify `luwcontrol=SERVER` in the linkage table. With this specification, each server call is a separate unit of work. The server program can issue commit or rollback requests as well.

Data Format Conversion

Code format conversion is performed on the server call as specified on the `contable` linkage table attribute and EZECONVT special function word. Code is converted based on the structure of the arguments specified on the CALL statement in the client program.

Error Handling

The standard error handling procedures described in “Communication Error Handling” on page 22 are supported with CICS ECI. EZERT8 is set to the decimal value of the client access service reason code. The reason code is the same as the number portion of error message CSOnnnna as listed in the *VisualAge Generator Messages and Problem Determination Guide* document.

Specifying Parameter Format

Parameter format (the `parmform` attribute on the calllink tag) can be specified as COMMDATA (data values passed in the COMMAREA) or COMMPTR (pointers passed in the COMMAREA).

Using COMMPTR Format: Use COMMPTR for calling COBOL subroutines that were developed to use the non-VAGen parameter passing convention that uses pointers in the CICS COMMAREA. When the pointer format is used, the generated client program calls an intermediate catcher program (ELACSV, provided with VisualAge Generator host services) via the ECI instead of calling the server program directly. The catcher program receives the parameter data from the client, builds a pointer list in the COMMAREA that points to the individual parameters, and links to the server program. A program processing table (PPT) must be defined for the ELACSV program on the CICS server system.

COMMPTR format is only supported for MVS CICS, CICS for VSE/ESA, and CICS OS/2 servers.

Using COMMDATA Format: Use COMMDATA for calling generated programs and for calling COBOL subroutines where possible. COMMDATA provides better performance in calling remote programs. With COMMDATA, the server program is called directly without going through the catcher program.

Configuring a Windows NT Client for a CICS for AIX Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─CICSCLIENT──────►

►──┬─────────────────────────────────────────┬──┬─────────────────────────────────┬──────►
   └─contable=─┬─conversion table name─┬─────┘  └─location=─┬─EZELOC──────┬──────┘
               ├─'*'───────────────────┤                   └─system name─┘
               ├─EZECONVT──────────────┤
               ├─NONE──────────────────┤
               └─BINARY────────────────┘

►──┬───────────────────────────────┬──┬──────────────────────────────┬───────────────────►
   │              ┌─CLIENT─┐       │  └─parmform=─┬─OSLINK───┬──────┘
   └─luwcontrol=──┴─SERVER─┴───────┘              ├─COMMDATA─┤
                                                  └─CICSOSLINK┘

►──┬───────────────────────────────┬──┬─────────────────────────────────┬────────────────►
   └─remoteapptype=─┬─VG────┬──────┘  │               ┌─GENERATION─┐    │
                    ├─NONVG─┤         └─remotebind=───┴─RUNTIME────┴────┘
                    └─ITF───┘

►──┬─────────────────────────────────────┬───────────────────────────────────────────►◄
   └─serverid=──server identifier────────┘
```

The following attributes are ignored:
- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST contable=BINARY
```

Configuring a Windows NT Client for a CICS for OS/2 Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─CICSCLIENT───────►


►─┬──────────────────────────────────────────┬─┬──────────────────────────────┬──────────►
  └─contable=──┬─conversion table name─┬─┘ └─location=──┬─EZELOC──────┬─┘
               ├─'*' ──────────────────┤                 └─system name─┘
               ├─EZECONVT──────────────┤
               ├─NONE──────────────────┤
               └─BINARY────────────────┘


►─┬────────────────────────────┬─┬──────────────────────────────┬────────────────────────►
  │              ┌─CLIENT─┐    │ └─parmform=──┬─OSLINK─────┬─┘
  └─luwcontrol=──┴─SERVER─┴────┘              ├─COMMPTR────┤
                                              ├─COMMDATA───┤
                                              └─CICSOSLINK─┘


►─┬────────────────────────────────┬─┬─────────────────────────────────────┬─────────────►
  └─remoteapptype=──┬─VG────────┬──┘ │                ┌─GENERATION─┐      │
                    ├─NONVG─┤        └─remotebind=────┴─RUNTIME────┴──────┘
                    └─ITF───┘


►─┬────────────────────────────────────┬─────────────────────────────────────────────────►◄
  └─serverid=───server identifier──────┘
```

The following attributes are ignored:
- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST
```

Configuring a Windows NT Client for a CICS for Windows NT Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─CICSCLIENT───────►
```

```
►─┬──────────────────────────────────────┬─┬───────────────────────────┬──────────►
  └─contable=─┬─conversion table name─┬─┘ └─location=─┬─EZELOC──────┬─┘
              ├─'*' ──────────────────┤                └─system name─┘
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬────────────────────────────┬─┬─────────────────────────────────┬──────────────►
  │              ┌─CLIENT─┐    │ └─parmform=─┬─OSLINK────┬─────────┘
  └─luwcontrol=──┴─SERVER─┴────┘             ├─COMMDATA──┤
                                             └─CICSOSLINK┘

►─┬──────────────────────────────┬─┬──────────────────────────────┬───────────────►
  └─remoteapptype=─┬─VG────┬─────┘ │                ┌─GENERATION─┐│
                   ├─NONVG─┤       └─remotebind=────┴─RUNTIME────┴┘
                   └─ITF───┘

►─┬──────────────────────────────────────┬────────────────────────────────────────►◄
  └─serverid=──server identifier─────────┘
```

The following attributes are ignored:
- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST
```

Configuring a Windows NT Client for an CICS for MVS/ESA Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─CICSCLIENT──────►

►─┬──────────────────────────────────────┬─┬───────────────────────────┬──────────►
  └─contable=─┬─conversion table name─┬─┘ └─location=─┬─EZELOC──────┬─┘
              ├─'*' ──────────────────┤                └─system name─┘
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘
```

```
►──┬──────────────────────────────┬──┬────────────────────────────────┬──────────────────────►
   │                ┌─CLIENT─┐    │  └─parmform=──┬─OSLINK─────┬──┘
   └─luwcontrol=────┴─SERVER─┴────┘               ├─COMMPTR────┤
                                                  ├─COMMDATA───┤
                                                  └─CICSOSLINK─┘

►──┬─────────────────────────────────┬──┬──────────────────────────────────┬─────────────────►
   └─remoteapptype=──┬─VG────────┬───┘  │                ┌─GENERATION─┐    │
                     ├─NONVG─────┤      └─remotebind=────┴─RUNTIME────┴────┘
                     └─ITF───────┘

►──┬──────────────────────────────────────────┬──────────────────────────────────────────►◄
   └─serverid=──server identifier─────────────┘
```

The following attributes are ignored:
- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST contable=ELACNENU
```

Configuring a Windows NT Client for a VSECICS Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─CICSCLIENT──────────►

►──┬────────────────────────────────────────────┬──┬────────────────────────────────────┬──►
   └─contable=──┬─conversion table name─────────┬┘  └─location=──┬─EZELOC──────────┬─────┘
                ├─'*'───────────────────────────┤                └─system name─────┘
                ├─EZECONVT──────────────────────┤
                ├─NONE──────────────────────────┤
                └─BINARY────────────────────────┘

►──┬──────────────────────────────┬──┬────────────────────────────────┬──────────────────────►
   │                ┌─CLIENT─┐    │  └─parmform=──┬─OSLINK─────┬──┘
   └─luwcontrol=────┴─SERVER─┴────┘               ├─COMMPTR────┤
                                                  ├─COMMDATA───┤
                                                  └─CICSOSLINK─┘
```

```
►──┬──────────────────────────────┬──┬─────────────────────────────────┬──────────────────►
   └─remoteapptype=──┬─VG────┬────┘  │                ┌─GENERATION─┐   │
                     ├─NONVG─┤       └─remotebind=────┴─RUNTIME────┴───┘
                     └─ITF───┘

►──┬────────────────────────────────────┬─────────────────────────────────────────────────►◄
   └─serverid=──server identifier───────┘
```

The following attributes are ignored:
- externalname
- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST contable=ELACNENU
```

Configuring a Windows NT Client for a CICS for Solaris Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call CICS transactions via the CICS ECI:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─CICSCLIENT──────►

►──┬───────────────────────────────────────────┬──┬─────────────────────────────────┬────►
   └─contable=──┬─conversion table name──┬──────┘  └─location=──┬─EZELOC───────┬─────┘
                ├─'*' ───────────────────┤                      └─system name──┘
                ├─EZECONVT───────────────┤
                ├─NONE───────────────────┤
                └─BINARY─────────────────┘

►──┬───────────────────────────────┬──┬────────────────────────────────────┬──────────────►
   │                ┌─CLIENT─┐     │  └─parmform=──┬─OSLINK─────┬─────────┘
   └─luwcontrol=────┴─SERVER─┴─────┘               ├─COMMDATA───┤
                                                   └─CICSOSLINK─┘

►──┬──────────────────────────────┬──┬─────────────────────────────────┬──────────────────►
   └─remoteapptype=──┬─VG────┬────┘  │                ┌─GENERATION─┐   │
                     ├─NONVG─┤       └─remotebind=────┴─RUNTIME────┴───┘
                     └─ITF───┘

►──┬────────────────────────────────────┬─────────────────────────────────────────────────►◄
   └─serverid=──server identifier───────┘
```

The following attributes are ignored:
- externalname

- library

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=CICSCLIENT
   location=CICSTST contable=BINARY
```

DCE Protocol

Processing Flow for VisualAge Generator DCE Common Services Remote Call

This section shows the processing flow for a VisualAge Generator DCE common services remote call.

1. VisualAge Generator DCE Server is started using a configuration file which specifies the DCE principal name that the server will obtain its DCE authorizations from (equivalent to a DCE userid), the location and the serverid name for binding information advertising, the Access Control List (ACL) object for client authorization, and the server programs that the server is authorized to process. The server programs are specified in one of two different groups; those in which secure DCE (authenticated RPC) communications are required and those which can be accessed via unsecured (unauthenticated RPC) DCE communications.

   The server obtains an object UUID for each program from the CDS object /.:/Servers/VAGenerator/SERVERID/program. If the program object is not defined, one will be created for it. The server uses the program object UUIDs when it advertises its binding information.
2. VisualAge Generator DCE Server advertises each of the programs that it services. The Cell Directory Services (CDS) location used for advertising the binding information is /.:/Servers/VAGenerator/SERVERID/LOCATION.
3. VisualAge Generator client retrieves the object UUID for the program from /.:/Servers/VAGenerator/SERVERID/program-name. It then uses the object UUID to request the binding information for a server which services the program from the CDS location /.:/Servers/VAGenerator/SERVERID/LOCATION. If there are multiple server bindings that match the search criteria, DCE will randomly return one of them.
4. VisualAge Generator client will setup for authenticated RPC, if DCESECURE is specified in the linkage table.
5. VisualAge Generator client performs data conversion on passed parameters as specified in the `contable` attribute of the client linkage table (runtime or generation time as applicable).
6. VisualAge Generator client makes remote call to VisualAge Generator DCE Server.

7. VisualAge Generator DCE Server checks if VisualAge Generator client is authorized to use server program (via DCE CDS Access Control List) and if the appropriate level of communication security is being used from the client.
8. VisualAge Generator DCE Server checks if the server program requested is one that it is authorized to process (via initial configuration file).
9. VisualAge Generator DCE Server processes client request, closes Logical Unit of Work, and returns data to client.

User Authentication and Authorization

User authentication is performed on the client via the DCE security server using the client's DCE login identifier. The VisualAge Generator DCE server checks whether the client is authorized to call the DCE server using DCE ACL security services. User authorization is performed via the DCE ACL security services. The VisualAge Generator DCE server is told at startup time the DCE object ACL to use for checking client authorization for running the called server programs. There is only one ACL used per VisualAge Generator DCE server; therefore, the authorization is at the VisualAge Generator DCE server level and not the called program level. If a called program requires a special ACL, then another VisualAge Generator DCE server will have to be created or started. The test ACL attribute determines whether or not the client is authorized to execute the server program (if the client has test privileges on the ACL object, then the client is authorized to execute all server programs provided by the server).

Controlling the Unit of Work

All calls via the DCE common services are server units of work. All resource changes are committed when the server returns to the calling program.

Data Format Conversion

Code format conversion is performed on the server call as specified on the `contable` attribute for the server entry in the linkage table. Data is converted based on the structure of the parameters specified on the server call in the client program.

Error Handling

The standard error handling procedures described in "Communication Error Handling" on page 22 are supported with VisualAge Generator DCE common services. EZERT8 is set to the decimal value of the client access service reason code. EZERT8 is only set when the REPLY option is coded on the call to the remote server program. If a visual link is used to make the call from a GUI program, the REPLY option is used on the call.

Any errors trapped by DCE are passed to the client program with a corresponding CSO error message. The error message contains an insert with the DCE mnemonic. A non-zero return code from the called program is passed back to the client program with a corresponding CSO error message. The error message contains an insert with the return code from the called program. VisualAge Generator return codes are documented in the help for the message.

All errors are traced to the CSO trace file on the client and server machines, as applicable.

Configuring a Windows NT Client for an OS/2 Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call server programs via the DCE common services:

```
►►──:calllink──applname=program name──linktype=─REMOTE──────────────────────────►

►─remotecomtype=──┬─DCE───────┬──┬──────────────────────┬────────────────────────►
                  └─DCESECURE─┘  └─parmform=─COMMDATA───┘

►──┬─────────────────────────────────────────┬──┬────────────────────────────┬──►
   └─contable=──┬─conversion table name─┬────┘  └─location=──┬─EZELOC──────┬─┘
                ├─'*' ──────────────────┤                    └─system name─┘
                ├─EZECONVT──────────────┤
                ├─NONE──────────────────┤
                └─BINARY────────────────┘

►──┬────────────────────────────────────┬──┬──────────────────────────────────┬──►◄
   │               ┌─GENERATION─┐       │  └─serverid=───server identifier───┘
   └─remotebind=───┴─RUNTIME────┴───────┘
```

The following attributes are ignored:
- remoteapptype
- externalname
- library
- luwcontrol

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=DCE
   location=Server1 remotebind=RUNTIME serverid=Test
```

Configuring a Windows NT Client for a Windows NT Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call server programs via the DCE common services:

```
►►──:calllink──applname=program name──linktype=─REMOTE─────────────────────────►

►─remotecomtype=─┬─DCE───────┬──┬───────────────────┬──────────────────────────►
                 └─DCESECURE─┘  └─parmform=─COMMDATA─┘

►──┬─────────────────────────────────────┬──┬──────────────────────────────┬───►
   └─contable=─┬─conversion table name─┬─┘  └─location=─┬─EZELOC──────┬─┘
               ├─'*' ──────────────────┤                └─system name─┘
               ├─EZECONVT──────────────┤
               ├─NONE──────────────────┤
               └─BINARY────────────────┘

►──┬────────────────────────────────┬──┬──────────────────────────────┬───────►◄
   │              ┌─GENERATION─┐    │  └─serverid=───server identifier─┘
   └─remotebind=──┴─RUNTIME────┴────┘
```

The following attributes are ignored:
- remoteapptype
- externalname
- library
- luwcontrol

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=DCE
   location=Server1 remotebind=RUNTIME serverid=Test
```

Configuring a Windows NT Client for an AIX Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes when you generate or test client programs that call server programs via the DCE common services:

```
►►──:calllink──applname=program name──linktype=─REMOTE─────────────────────────►

►─remotecomtype=─┬─DCE───────┬──┬───────────────────┬──────────────────────────►
                 └─DCESECURE─┘  └─parmform=─COMMDATA─┘
```

```
►─┬──────────────────────────────────────────┬─┬──────────────────────────────┬──────────────►
  └─contable=─┬─conversion table name─┬──────┘ └─location=─┬─EZELOC──────┬───┘
              ├─'*' ──────────────────┤                     └─system name─┘
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘


►─┬──────────────────────────────────┬─┬────────────────────────────────────┬──────────────►◄
  │                ┌─GENERATION─┐    │ └─serverid=──server identifier──────┘
  └─remotebind=────┴─RUNTIME────┴────┘
```

The following attributes are ignored:
- remoteapptype
- externalname
- library
- luwcontrol

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=DCE
   location=Server1 contable=BINARY remotebind=RUNTIME serverid=Test
```

DIRECT Protocol

Advantages of IPC and DIRECT Protocols

The protocols IPC and DIRECT provide a method for client programs to call server programs where both reside on the same system. A local call from the client program to the server program might be used, but there are advantages to making this call remotely via the VisualAge Generator middleware in certain cases. Futhermore, these advantages differ for the provided protocols, IPC and DIRECT, and are described below.

Calls to local C++ applications for database access from Smalltalk GUI applications where the GUI application might run in the same session with other unrelated GUI applications that require database access: GUI applications begin VisualAge Generator communications sessions when calling server programs. When the IPC protocol is used, each call from a different GUI would be made under a different process, each of which can have its own database connection and unit of work. This would not be possible if the client program used a local call to access the server program.

If two GUI applications need to be in the same unit of work, start the second GUI from the first and define the second GUI with the VAGen inheritsCommSession attribute set to true in the AbtAppBldrPart or AbtAppBldrView class. This way, local servers called by both GUI applications will share the same communications session, database connection, and unit of work.

Calls to generated C++ application from Java clients where the C++ application is to run on a web server: Multiple calls might arrive simultaneously, thus the need for multiple, independent server-controlled units of work provided by the IPC protocol.

When using the DIRECT protocol, the call is similar to a local call directly from a client program to a server program except that the VisualAge Generator middleware performs the call. Because no separate process is created, the DIRECT protocol provides improved performance over the IPC protocol. The following are scenarios for using the DIRECT protocol:

Calls to a server program running under the control of VisualAge Generator test facility from Java, VisualBasic, or PowerBuilder, where the client and server resides on the same system.

Calls to a server program running under the control of Component Broker, which maintains the database connection itself. In such an environment, the server program should not run under a separate process so that the database connection cannot be inadvertently closed when the process ends.

Error Handling

The standard error handling procedures described in "Communication Error Handling" on page 22 are supported with IPC and DIRECT server calls. EZERT8 is set to the decimal value of the client access service reason code. The reason code is the same as the number portion of error message CSOnnnna as listed in the *VisualAge Generator Messages and Problem Determination Guide* document.

Configuring a Windows NT Client for an Windows NT Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via DIRECT:

```
►►—:calllink—applname=program name—linktype=–CSOCALL—remotecomtype=–DIRECT————————►◄
```

While the specification of:

```
►►—contable=—┬—conversion table name—┬———————————————————————————————————————————►◄
             ├—'*' ——————————————————┤
             ├—EZECONVT——————————————┤
             ├—NONE——————————————————┤
             └—BINARY————————————————┘
```

is permissible, data conversion is generally not necessary when the client and server reside on the same system. All other linkage table entries are ignored on a local (DIRECT) call.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=CSOCALL remotecomtype=DIRECT
```

IPC Protocol

Advantages of IPC and DIRECT Protocols

The protocols IPC and DIRECT provide a method for client programs to call server programs where both reside on the same system. A local call from the client program to the server program might be used, but there are advantages to making this call remotely via the VisualAge Generator middleware in certain cases. Futhermore, these advantages differ for the provided protocols, IPC and DIRECT, and are described below.

Calls to local C++ applications for database access from Smalltalk GUI applications where the GUI application might run in the same session with other unrelated GUI applications that require database access: GUI applications begin VisualAge Generator communications sessions when calling server programs. When the IPC protocol is used, each call from a different GUI would be made under a different process, each of which can have its own database connection and unit of work. This would not be possible if the client program used a local call to access the server program.

If two GUI applications need to be in the same unit of work, start the second GUI from the first and define the second GUI with the VAGen inheritsCommSession attribute set to true in the AbtAppBldrPart or AbtAppBldrView class. This way, local servers called by both GUI applications will share the same communications session, database connection, and unit of work.

Calls to generated C++ application from Java clients where the C++ application is to run on a web server: Multiple calls might arrive simultaneously, thus the need for multiple, independent server-controlled units of work provided by the IPC protocol.

When using the DIRECT protocol, the call is similar to a local call directly from a client program to a server program except that the VisualAge Generator middleware performs the call. Because no separate process is created, the DIRECT protocol provides improved performance over the IPC protocol. The following are scenarios for using the DIRECT protocol:

Calls to a server program running under the control of VisualAge Generator test facility from Java, VisualBasic, or PowerBuilder, where the client and server resides on the same system.

Calls to a server program running under the control of Component Broker, which maintains the database connection itself. In such an environment, the server program should not run under a separate process so that the database connection cannot be inadvertently closed when the process ends.

Error Handling

The standard error handling procedures described in "Communication Error Handling" on page 22 are supported with IPC and DIRECT server calls. EZERT8 is set to the decimal value of the client access service reason code. The reason code is the same as the number portion of error message CSOnnnna as listed in the *VisualAge Generator Messages and Problem Determination Guide* document.

Configuring a Windows NT Client for an Windows NT Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via IPC:

```
►►──:calllink──applname=program name──linktype=─CSOCALL──remotecomtype=─IPC──────────►◄
```

While the specification of:

```
►►──contable=─┬─conversion table name─┬──────────────────────────────────────────────►◄
              ├─'*' ──────────────────┤
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘
```

is permissable, data conversion is generally not necessary when the client and server reside on the same system. All other linkage table entries are ignored on a local (IPC) call.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=CSOCALL remotecomtype=IPC
```

TCP/IP Protocol

Creating a TCP/IP Services File Entry

The client machine must have an entry for the TCP/IP Serverid added to its TCP/IP services file. For Windows, AIX, HP-UX, and Solaris the TCP/IP SERVICES file is used.

Error Handling

The standard error handling procedures described in “Communication Error Handling” on page 22 are supported with TCPIP server calls. EZERT8 is set to the decimal value of the client access service reason code. The reason code is the same as the number portion of error message CSOnnnna as listed in the *VisualAge Generator Messages and Problem Determination Guide* document.

Configuring a Windows NT Client for an OS/2 Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─TCPIP──────────►

►──┬──────────────────────────────────────┬──────────────────────────────────────────►
   └─contable=─┬─conversion table name─┬──┘
               ├─'*' ──────────────────┤
               ├─EZECONVT──────────────┤
               ├─NONE──────────────────┤
               └─BINARY────────────────┘

►──┬─────────────────────────────────────────────┬───────────────────────────────────►
   └─location=───system name_tcpip_hostname──────┘

►──┬────────────────────────────────────────┬───────────────────────────────────────►◄
   └─serverid=───tcpip_server identifier────┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to “Conversion Table by Language and Platform” on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 remotebind=RUNTIME serverid=port1
```

Configuring a Windows NT Client

Configuring a Windows NT Client for a Windows NT Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─TCPIP──────────►

►─┬─────────────────────────────────────┬──────────────────────────────────────────────►
  └─contable=─┬─conversion table name─┬─┘
              ├─'*' ──────────────────┤
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬──────────────────────────────────────────┬─────────────────────────────────────────►
  └─location=───system name_tcpip_hostname───┘

►─┬────────────────────────────────────────┬───────────────────────────────────────────►◄
  └─serverid=───tcpip_server identifier────┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to "Conversion Table by Language and Platform" on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 remotebind=RUNTIME serverid=port1
```

Configuring a Windows NT Client for an AIX Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─TCPIP──────────►
```

```
►─┬────────────────────────────────────┬───────────────────────────────────────────►
  └─contable=─┬─conversion table name─┬─┘
              ├─'*' ──────────────────┤
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘
```

```
►─┬──────────────────────────────────────────┬─────────────────────────────────────►
  └─location=──system name_tcpip_hostname────┘
```

```
►─┬───────────────────────────────────────┬───────────────────────────────────────►◄
  └─serverid=──tcpip_server identifier────┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to “Conversion Table by Language and Platform” on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 contable=BINARY remotebind=RUNTIME serverid=port1
```

Configuring a Windows NT Client for an HP-UX Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─TCPIP──────────►
```

```
►─┬────────────────────────────────────┬───────────────────────────────────────────►
  └─contable=─┬─conversion table name─┬─┘
              ├─'*' ──────────────────┤
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘
```

```
►─┬──────────────────────────────────────────┬─────────────────────────────────────►
  └─location=──system name_tcpip_hostname────┘
```

```
►─┬──────────────────────────────────┬─────────────────────────────────────►◄
  └─serverid=──tcpip_server identifier─┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to "Conversion Table by Language and Platform" on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 contable=BINARY remotebind=RUNTIME serverid=port1
```

Configuring a Windows NT Client for a Solaris Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►─:calllink─applname=program name─linktype=─REMOTE─remotecomtype=─TCPIP──────────►

►─┬───────────────────────────────────┬─────────────────────────────────────►
  └─contable=─┬─conversion table name─┬─┘
              ├─'*' ──────────────────┤
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬──────────────────────────────────────────┬──────────────────────────────►
  └─location=──system name_tcpip_hostname────┘

►─┬──────────────────────────────────┬─────────────────────────────────────►◄
  └─serverid=──tcpip_server identifier─┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to "Conversion Table by Language and Platform" on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 contable=BINARY remotebind=RUNTIME serverid=port1
```

Configuring a Windows NT Client for a VM/ESA Server

Linkage Table Attributes for Generating Client Programs: Specify the following calllink attributes for the server program when you generate or test clients that call the server program via TCPIP:

```
►►──:calllink──applname=program name──linktype=─REMOTE──remotecomtype=─TCPIP──────────►

►──┬────────────────────────────────────┬──────────────────────────────────────────────►
   └─contable=──┬─conversion table name─┬┘
                ├─'*' ──────────────────┤
                ├─EZECONVT──────────────┤
                ├─NONE──────────────────┤
                └─BINARY────────────────┘

►──┬──────────────────────────────────────────┬────────────────────────────────────────►
   └─location=───system name_tcpip_hostname───┘

►──┬───────────────────────────────────────┬───────────────────────────────────────────►◄
   └─serverid=───tcpip_server identifier───┘
```

The entries specified for location and serverid are case sensitive.

The `contable` attribute must be specified for certain language and platform combinations. Refer to “Conversion Table by Language and Platform” on page 436 to help you determine if a conversion table is necessary, and which table should be specified.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=TCPIP
   location=server1 contable=ELACNENU remotebind=RUNTIME serverid=port1
```

Configuring a Windows NT Server

Summary Table of Valid Clients and Protocols

Table 32. Valid Clients and Protocols for VisualAge Generator Servers on the Windows NT Platform

Server Platform	**Client Platforms**				
	OS/2 (GUI, C++, ITF)	**Windows 95 and Windows 98 (GUI)**	**Windows NT (GUI, C++, ITF)**	**Windows NT (Java GUI, Java server)**	**AIX (C++)**
Windows NT (C++)	TCP/IP, DCE	TCP/IP, DCE	TCP/IP, DCE, IPC, DIRECT	TCP/IP, DCE, IPC, DIRECT	TCP/IP, DCE
Windows NT (Java)				TCP/IP, DIRECT	

DCE Protocol

List of Valid Clients

- OS/2 (GUI, C++, ITF)
- Windows 95 and Windows 98 (GUI)
- Windows NT (GUI, C++, ITF)
- AIX (C++)
- Solaris

Identifying the Server Location

The server location is determined by the bindings advertised in the DCE CDS database. The bindings are found in the DCE CDS location object located at /.:/Servers/VAGenerator/SERVERID/LOCATION. Identify the server location to the client by specifying the location identifier in the `location` linkage table attribute in the entry for the server program, or set the location dynamically in the client program at runtime using EZELOC.

Linkage Table Attributes for Generating Server Programs

To generate native OS/2, AIX, or Windows NT server programs, no linkage table entry is required. If the server program is going to make remote calls (it is the second tier of a three tier client/server system) then a linkage table needs to be provided at generation time just as if it was a client program.

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE remotecomtype=DCE
   location=Server1 remotebind=GENERATION serverid=Test
```

DCE CDS Entries Required for VisualAge Generator DCE Servers

All VisualAge Generator DCE CDS entries are placed under the /.:/Servers/VAGenerator directory. Each serverid must have a directory, /.:/Servers/VAGenerator/serverid.

Note: It might be helpful to equate serverid with an application system and location with a server or group of servers that process requests for the application system.

Starting the DCE Server Program

The VisualAge Generator DCE server program, CSODCES, is started on the remote host machine and listens for incoming DCE requests and processes them. On AIX systems, the current userid must either be root or have access to the DCE keytab file. CSODCES takes one optional parameter and one required parameter at startup. The required parameter is the name of the configuration file to be used in starting up the DCE server. The optional parameter specifies the type of cleanup the server does when it is terminated.

```
csodces [-c | -d ] filename
```

The configuration file contains the following:

- The principal name to be used by the VisualAge Generator DCE server for DCE access and authorization
- The advertising location and serverid names of the DCE server (specified in the `location` and `serverid` attributes of the client's linkage table)
- The DCE ACL object to be used for client authorization
- The called server programs that it will be processing requests for (specified in the applname attribute of the client's linkage table).

The programs are divided into those that require authorization checking (SECURE PROGRAMS) and those that do not require authorization checking (PUBLIC PROGRAMS). Authorization checking has an impact on server performance, so only use authorization when required.

The following example shows the configuration file:

```
DCEprincipal=vgserve1
LOCATION=Server1
SERVERID=Test
DCEACLobject=/.:/Servers/VAGenerator/Test/Server1
SECURE PROGRAMS=
SECURE1
SECURE2
PUBLIC PROGRAMS=
ELACVP5
DTCALL2
MAXSIZE
PARMSRV
PRMSRV2
```

The cleanup parameter is used when there is more than one server using a serverid/location pair as its advertising location. With the -c option, which is the default, when the server terminates it will remove its entry from the RPC mapping, DCE runtime, and DCE CDS. If there are multiple servers advertising at a serverid/location location and one of the servers removes its entry, then all of the servers will lose their entries. To prevent all the servers from losing their entries, use the -d parameter which will only remove the entry from the RPC mapping when the server terminates. There should always be one server which is started without the -d parameter to ensure that all entries for the DCE servers are cleaned up after they are terminated. The server that was started without the -d parameter should be terminated last.

DIRECT Protocol

List of Valid Clients

- Windows NT (GUI, C++, ITF, Java)

Identifying a C++ Server Location

When using the IPC or DIRECT protocols, server programs are located using the LIBPATH environment variable. You need to ensure that the server DLL directory is specified in the LIBPATH.

Identifying a Java Server Location

When you use the DIRECT protocol, Java server programs are located using the CLASSPATH environment variable. This variable also contains the path to the properties files associated with the server program.

Java Server Program Example Linkage

VisualAge Generator Java server programs can call other VAGen server programs locally or remotely. Settings for calls between server programs can be specified in a VAGen Linkage Table and generated into a properties file. The properties file, used by the Java server program at run time, can also be created manually. If the linkage table is accessed at runtime, some defaults from the properties file may be assumed. For additional information, see “Calls between Server Programs and VAGen Java Server Programs” on page 18.

VAGen Java Server Program Calling a C++ Server Program in a DLL on the Same System: The following examples illustrate how linkage from a VAGen Java server program to a VAGen C++ server program in a DLL on the same system is specified in the linkage table for a generation-time bind or a run-time bind. Note that the properties file used during a run-time bind contains several default properties not specified in the linkage table entry.

```
:calllink applname=SERVER linktype=CSOCALL remotecomtype=DIRECT contable=CSOI1252
```

Figure 2. Linkage Table Entry — Generation-time Bind

Figure 3 shows an example of a linkage table entry for a run-time bind. Figure 4 shows the properties as they would appear in the Java properties file.

```
:calllink applname=SERVER linktype=CSOCALL remotebind=RUNTIME
```

Figure 3. Linkage Table Entry — Run-time Bind

```
cso.serverLinkage.SERVER.remotecomtype=DIRECT
cso.serverLinkage.SERVER.contable=CSOI1252
```

Figure 4. Java Properties File Entry

VAGen Java Server Program Starting a Java Web Transaction on the Same System: The following examples illustrate how linkage from a VAGen Java server program to a local Java web transaction is specified. The package of the bean and the web transaction is my.pkg.

Note: These properties are entered in the GatewayServlet's properties file. They are not the same as the linkage table properties used by Java servers.

Figure 5 shows the properties as they would appear in the Java properties file.

```
serverLinkage.SERVER.commtype=DIRECT
serverLinkage.SERVER.javaProperty=my.pkg
serverLinkage.SERVER.remoteapptype=VGJAVA
```

Figure 5. GatewayServlet Properties File Entry

IPC Protocol

List of Valid Clients

- Windows NT (GUI, C++, ITF)

Identifying the Server Location

When using the IPC or DIRECT protocols, server programs are located using the LIBPATH environment variable. You need to ensure that the server DLL directory is specified in the LIBPATH.

TCP/IP Protocol

List of Valid Clients

- OS/2 (GUI, C++, ITF)
- Windows 95 and Windows 98 (GUI)
- Windows NT (GUI, C++, ITF, Java)
- AIX (C++)

Identifying a C++ Server Location

The TCP/IP support servers locate the server programs via the LIBPATH environment variable, so you need to ensure that the server DLL directory is specified in the LIBPATH.

Identifying a Java Server Location

When you use TCP/IP, Java server programs are located using the CLASSPATH environment variable. This variable also contains the path to the properties files associated with the server program.

Java Server Program Example Linkage

VisualAge Generator Java server programs can call other VAGen server programs locally or remotely. Settings for calls between server programs can be specified in a VAGen Linkage Table and generated into a properties file. The properties file, used by the Java server program at run time, can also be created manually. If the linkage table is accessed at runtime, some defaults from the properties file may be assumed. For additional information, see "Calls between Server Programs and VAGen Java Server Programs" on page 18.

VAGen Java Server Program Calling a Remote Java Server Program: The following examples illustrate how linkages between remote Java server programs are specified in the linkage table for a generation-time bind or a run-time bind. Only required properties are shown Note that the properties file used during a run-time bind contains several default properties not specified in the linkage table entry.

Figure 6 shows an example of a linkage table entry for a generation-time bind.

```
:calllink applname=SERVER linktype=CSOCALL remotecomtype=TCPIP location=ntjserv
serverid=9876 package='my.pkg' remoteapptype=VGJAVA
```

Figure 6. Linkage Table Entry — Generation-time Bind

Note: Linkage table entries should be on one line.

Figure 7 shows an example of a linkage table entry for a run-time bind of a Java server calling another Java server remotely over TCPIP. The remote server is on a machine called ntjserv, at port 9876. The called server's package is my.pkg. Figure 8 on page 269 shows the properties as they would appear in the Java properties file.

```
:calllink applname=SERVER linktype=CSOCALL remotebind=RUNTIME
```

Figure 7. Linkage Table Entry — Run-time Bind

```
cso.serverLinkage.SERVER.remotecomtype=TCPIP
cso.serverLinkage.SERVER.location=ntjserv
cso.serverLinkage.SERVER.serverid=9876
cso.serverLinkage.SERVER.package=my.pkg
cso.serverLinkage.SERVER.remoteapptype=VGJAVA
```

Figure 8. Java Properties File Entry

VAGen Java Server Program Calling a Remote C++ Server Program on AIX: The following examples illustrate how linkages between remote VAGen Java and C++ server programs are specified in the linkage table for a generation-time bind or a run-time bind. Note that the properties file used during a run-time bind contains several default properties not specified in the linkage table entry.

Figure 9 shows an example of a linkage table entry for a generation-time bind of a Java server calling a C++ server on AIX remotely over TCPIP. The remote server is on a machine called ntjserv, at port 9876.

```
:calllink applname=SERVER linktype=CSOCALL remotecomtype=TCPIP contable=CSOX850 location=ntjser
```

Figure 9. Linkage Table Entry — Generation-time Bind

Figure 10 shows an example of a linkage table entry for a run-time bind. Figure 11 shows the properties as they would appear in the Java properties file.

```
:calllink applname=SERVER linktype=CSOCALL remotebind=RUNTIME
```

Figure 10. Linkage Table Entry — Run-time Bind

```
cso.serverLinkage.SERVER.remotecomtype=TCPIP
cso.serverLinkage.SERVER.contable=CSOX850
cso.serverLinkage.SERVER.location=ntjserv
cso.serverLinkage.SERVER.serverid=9876
```

Figure 11. Java Properties File Entry

VAGen Java Server Program Starting a Remote Java Web Transaction: The following example illustrates how linkage from a VAGen Java server program to a remote Java web transaction is specified. The Web transaction in these examples is on a machine called ntjserv, at port 9876, and is using codepage 1252 (Windows NT English). The package of the bean and the web transaction is my.pkg.

Note: These properties are entered in the GatewayServlet's properties file. They are not the same as the linkage table properties used by Java servers.

Figure 12 shows the properties as they would appear in the GatewayServlet properties file.

```
serverLinkage.SERVER.commtype=TCPIP
serverLinkage.SERVER.contable=CSOJ1252
serverLinkage.SERVER.location=ntjserv
serverLinkage.SERVER.serverid=9876
serverLinkage.SERVER.javaProperty=my.pkg
serverLinkage.SERVER.remoteapptype=VGJAVA
```

Figure 12. GatewayServlet Properties File Entry

C++ Server Program Set Up and Operation

The server uses a configuration file for specifying the TCP/IP service name to listen on. The configuration file is optional as there are predefined default values that will be used. The configuration file can also be used to modify a ″performance″ parameter, tcp_start_process.

Start the TCP/IP server by issuing the command:

```
   CSOTCPS "config_filename"
```

The configuration file is located via the following search order:
1. File specified on the command line
2. The file named CSO.INI in the directory specified by the CSODIR environment variable.
3. The file named CSO.INI in the current directory

The search ends when the first one of the above conditions is met. Once a file is identified, the contents of the file are used if possible. If the file cannot be opened for any reason, a warning message is printed on the console and the default values are used. CSOTCPS will display the configuration filename and values being used prior to starting to listen for incoming requests. The default values are:

```
   TCP/IP service name: VAGenerator
   tcp_start_process: 4
```

Where:

TCP/IP service name	Specifies the service name that will be used to look up the TCP/IP port number that CSOTCPS will listen on for incoming requests.

	This entry is case sensitive and must match an entry in the TCP/IP services file. For Windows NT, AIX, and HP-UX the TCP/IP SERVICES file is used.
tcp_start_process	Specifies the number of server processes which will be prestarted. This number must be at least 1. Very lightly loaded systems could get away with specifying a lower value if you needed to minimize the number of running processes. Heavily loaded systems could see a moderate performance gain by increasing this value.

Sample TCP/IP Entries from CSO.INI File: The following TCP/IP entries are located in the sample CSO.INI file. All entries must start in column 1. Comments are allowed within the file by placing a semicolon (;) in column 1 of the comment line. You only need to specify an entry if you wish to override the default value. If duplicate entries are specified in the file, the last entry in the file is used (just like environment variables in the config.sys file).

```
tcp_service_name=VAGenerator
tcp_start_process=2
```

Java Server Program Set Up and Operation

To make remote calls using TCP/IP, the Java server program must have access to a default properties file. The path to that properties file can be specified on the command line or you can use the default path: `.\tcpiplistener.properties`.

To start the TCP/IP server, from a command line, run the following command:

```
java CSOTcpipListener
```

Where:

java Is the command used to start the JVM.

CSOTcpipListener
Is a Java program that handles TCP/IP calls.

For more information about using the listener properties or linkage properties, see “Appendix A. Java properties” on page 401. For more information on using other Java properties, see the *VisualAge Generator Generation Guide*.

Chapter 17. CICS for Windows NT Platform

How to use this chapter

This chapter contains platform and protocol information necessary to configure a client program or a server program. The chapter is divided into configuration sections: client programs and server programs. Each configuration section is organized by protocol. Each protocol section contains information organized by target platforms.

Use the following method to locate the information you need:

1. Verify that this chapter is appropriate to your runtime environment.

   Ensure that the runtime environment of your program matches the platform on which this chapter is based. Turn to the appropriate chapter if the runtime environment of your program does not match the platform named in the title of this chapter.
2. Determine which type of program you want to configure: client program or server program.
3. Determine which configuration section in the chapter is appropriate for your program.
4. Find the intended target platform of the program you are configuring in the Summary Table of Valid Clients and Protocols or Summary Table of Valid Servers and Protocols.

   If you are configuring a client program, find the target platform of the intended server program in the column of valid platforms.

   If you are configuring a server program, find the target platform of the intended client program in the row of valid platforms.
5. Chose a protocol based on the information in the table.

   If you are configuring a client program, find the cell in the table where the row of the intended target platform intersects the column of the client platform. Select a protocol from the list of available protocols for this client/server combination.

   If you are configuring a client program, find the cell in the table where the column of the intended target platform intersects the row of the server platform. Select a protocol from the list of available protocols for this client/server combination.
6. Determine the name of the appropriate protocol section within the configuration section.

Turn to the part of the configuration section indicated by the protocol name. Here you will find information common to all target platforms valid for this protocol.

7. Determine the name of the appropriate target platform section within the protocol section.

   You will find platform specific information for the chosen protocol in the section named for your intended target platform.

Example

```
Chapter: CICS for Windows NT Platform
Type of program to configure: Client program
Configuration section: Configuring a CICS for Windows NT client
Intended target platform: CICS for MVS/ESA (i.e. a CICS for MVS/ESA server program)
Chosen protocol: CICS DPL
Protocol section: CICS DPL Protocol
Target platform section: Configuring a CICS for Windows NT Client for a
CICS for MVS/ESA Server
```

You will encounter syntax diagrams in this chapter. The diagrams are used to show programming syntax as follows:

```
►►──:RequiredTerm──RequiredTerm=RequiredValue──RequiredTerm=──┬─OptionalValue─┬──────────►
                                                              └─OptionalValue─┘

►──┬──────────────────────────────────────┬──┬────────────────────────────────┬──────────►
   │                    ┌─DefaultValue──┐ │  └─OptionalTerm=RequiredValue──┘
   └─OptionalTerm=──────┴─OptionalValue─┴─┘

►──┬──────────────────────────────────────┬──────────────────────────────────────────────►◄
   └─OptionalTerm=──────┬─OptionalValue─┬─┘
                        └─OptionalValue─┘
```

RequiredTerm	A required term is a term that must be specified. Required terms are depicted in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required terms.
RequiredValue	A required value is a value that must be specified for a term. A required value is shown in italics in syntax diagrams on the same line as the term preceding it. There are no paths depicted directly around required values.
OptionalTerm	An optional term is a term that can be

	specified. An optional term is depicted in syntax diagrams in an alternate path from the preceding term.
OptionalValue	An optional value is a value that can be specified for a term. An optional value is shown in italics in syntax diagrams in an alternate path from the preceding term along with other optional values or a default value.
DefaultValue	A default value is the value that is specified by default for a term. A default value is shown in italics in syntax diagrams in an alternate path above other optional values.

Configuring a CICS for Windows NT Client

Summary Table of Valid Servers and Protocols

Table 33. Valid Servers and Protocols for VisualAge Generator Clients on the CICS for Windows NT Platform

Server Platforms	Protocols for CICS for Windows NT Client Platform
CICS for AIX	CICS DPL
CICS for MVS/ESA	CICS DPL
CICS for OS/2	CICS DPL
CICS for Solaris	CICS DPL
CICS for VSE/ESA	CICS DPL
CICS for Windows NT	CICS DPL

CICS DPL Protocol

User Authentication

The user associated with the CICS client transaction must be authorized to run the client transaction and the server transaction. The user ID and password are identified by the CICS transaction manager on the client system.

Setting Up Communication Links

The communication link between the client and server systems must be defined to the CICS systems to allow a distributed program link from one system to the other. Refer to the CICS intercommunication manual for your CICS systems for more information.

Controlling the Unit of Work

Use the `luwcontrol` linkage table attribute to indicate whether the server updates are automatically committed on return or the client controls the unit of work. If the client is controlling the unit of work, subsequent server calls must go to the same system and transaction under client-controlled unit of work until a commit or roll back is requested. The server cannot issue EZECOMIT or EZEROLLB calls if client-controlled unit of work is specified.

Data Format Conversion

Code format conversion is performed on the server call as specified in the `contable` linkage attribute table and in EZECONVT. Code is converted based on the structure of the arguments that are specified on the CALL statement in the client program.

Error Handling

The standard error handling procedures described below are supported with CICS DPL. If the DPL is not successful for any reason, including unavailability of the communication link, error information is returned with CICS in the CICS EIB (EXEC interface block).

If the REPLY option is not specified on the CALL statement, the calling program ends with error messages. If REPLY is specified, no messages are written or logged, and EZERT8 is set to one of the following values:

Value	Meaning
00000000	Successful call and return
00000204	Program name not valid
00000207	System identifier not valid
00000208	Link out of service
ffrrrrrr	Other CICS error where ff is the hexadecimal representation of EIBFN byte 0, and rrrrrr is the hexadecimal representation of EIBRCODE bytes 0–2.

Configuring a CICS for Windows NT Client for a CICS for OS/2 Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►—:calllink—applname=program name—linktype=—REMOTE—┬──────────────────────┬——————►
                                                     └─parmform=—COMMDATA───┘
```

```
►──┬─────────────────────────────────────────┬─────────────────────────────────────────►
   │                ┌─GENERATION─┐           │
   └─remotebind=────┴────────────┴─RUNTIME───┘

►──┬───────────────────────────────────────┬──┬──────────────────────────────┬──────────►
   └─contable=──┬─conversion table name─┬──┘  └─location=──┬─EZELOC──────┬──┘
                ├─'*' ──────────────────┤                  └─system name─┘
                ├─EZECONVT──────────────┤
                ├─NONE──────────────────┤
                └─BINARY────────────────┘

►──┬─────────────────────────────┬──┬──────────────────────────────┬────────────────────►
   │                ┌─CLIENT─┐   │  └─remoteapptype=──┬─VG────┬────┘
   └─luwcontrol=────┴─SERVER─┴───┘                    ├─NONVG─┤
                                                      └─ITF───┘

►──┬─────────────────────────────────────┬──────────────────────────────────────────────►◄
   └─serverid=───server identifier───────┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CS07 LUWCONTROL=CLIENT
```

Configuring a CICS for Windows NT Client for a CICS for Windows NT Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►──:calllink──applname=program name──linktype=─REMOTE──┬──────────────────────┬────────►
                                                        └─parmform=─COMMDATA───┘

►──┬─────────────────────────────────────────┬─────────────────────────────────────────►
   │                ┌─GENERATION─┐           │
   └─remotebind=────┴────────────┴─RUNTIME───┘
```

```
►─┬──────────────────────────────────────┬──┬──────────────────────────────┬──────────►
  └─contable=─┬─conversion table name─┬──┘  └─location=─┬─EZELOC──────┬─┘
              ├─'*' ──────────────────┤                 └─system name─┘
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬───────────────────────────────┬─┬─────────────────────────────┬──────────────────►
  │               ┌─CLIENT─┐      │ └─remoteapptype=─┬─VG────┬───┘
  └─luwcontrol=───┴─SERVER─┴──────┘                  ├─NONVG─┤
                                                     └─ITF───┘

►─┬─────────────────────────────────────┬───────────────────────────────────────────►◄
  └─serverid=──server identifier────────┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CSO7 LUWCONTROL=CLIENT
```

Configuring a CICS for Windows NT Client for a CICS for AIX Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►──:calllink──applname=program name──linktype=─REMOTE──┬──────────────────────┬────►
                                                        └─parmform=─COMMDATA───┘

►─┬────────────────────────────────────────┬────────────────────────────────────────►
  │             ┌─GENERATION─┐             │
  └─remotebind=─┴────────────┴─RUNTIME─────┘

►─┬──────────────────────────────────────┬──┬──────────────────────────────┬──────────►
  └─contable=─┬─conversion table name─┬──┘  └─location=─┬─EZELOC──────┬─┘
              ├─'*' ──────────────────┤                 └─system name─┘
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘
```

```
►──┬─────────────────────────────┬──┬──────────────────────────────┬──────────────►
   │              ┌─CLIENT──┐    │  └─remoteapptype=──┬─VG─────┬──┘
   └─luwcontrol=──┴─SERVER──┴────┘                    ├─NONVG──┤
                                                      └─ITF────┘

►──┬──────────────────────────────────┬───────────────────────────────────────────►◄
   └─serverid=──server identifier─────┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CS07 contable=BINARY LUWCONTROL=CLIENT
```

Configuring a CICS for Windows NT Client for a CICS for MVS/ESA Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►──:calllink──applname=program name──linktype=─REMOTE──┬─────────────────────┬───►
                                                        └─parmform=─COMMDATA──┘

►──┬─────────────────────────────────────────┬────────────────────────────────────►
   │             ┌─GENERATION─┐              │
   └─remotebind=─┴────────────┴──RUNTIME─────┘

►──┬──────────────────────────────────────┬──┬───────────────────────────┬────────►
   └─contable=──┬─conversion table name──┬┘  └─location=──┬─EZELOC───────┬┘
                ├─'*'────────────────────┤                └─system name──┘
                ├─EZECONVT───────────────┤
                ├─NONE───────────────────┤
                └─BINARY─────────────────┘

►──┬─────────────────────────────┬──┬──────────────────────────────┬──────────────►
   │              ┌─CLIENT──┐    │  └─remoteapptype=──┬─VG─────┬──┘
   └─luwcontrol=──┴─SERVER──┴────┘                    ├─NONVG──┤
                                                      └─ITF────┘
```

```
►─┬──────────────────────────────┬─────────────────────────────────────────►◄
  └─serverid=──server identifier─┘
```

The following attributes are ignored:
- externalname
- library
- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CS07 contable=ELACNENU LUWCONTROL=CLIENT
```

Configuring a CICS for Windows NT Client for a CICS for VSE/ESAServer

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►─:calllink─applname=program name─linktype=─REMOTE─┬──────────────────────┬───►
                                                    └─parmform=─COMMDATA───┘

►─┬────────────────────────────────────────┬───────────────────────────────────►
  │                ┌─GENERATION─┐          │
  └─remotebind=────┴────────────┴─RUNTIME──┘

►─┬─────────────────────────────────────────┬─┬────────────────────────────┬───►
  └─contable=─┬─conversion table name─┬─────┘ └─location=─┬─EZELOC──────┬─┘
              ├─'*'───────────────────┤                  └─system name─┘
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬────────────────────────────┬─┬──────────────────────────┬──────────────────►
  │              ┌─CLIENT─┐    │ └─remoteapptype=─┬─VG────┬─┘
  └─luwcontrol=──┴─SERVER─┴────┘                  ├─NONVG─┤
                                                  └─ITF───┘

►─┬──────────────────────────────┬─────────────────────────────────────────►◄
  └─serverid=──server identifier─┘
```

The following attributes are ignored:
- externalname
- library

- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CS07 contable=ELACNENU LUWCONTROL=CLIENT
```

Configuring a CICS for Windows NT Client for a CICS for Solaris Server

Linkage Table Attributes for Generating CICS Client Programs: Specify the following calllink attributes for the server program when you generate CICS client programs that call the server via CICS DPL:

```
►►─:calllink─applname=program name─linktype=─REMOTE─┬─────────────────────┬───────►
                                                    └─parmform=─COMMDATA──┘

►─┬──────────────────────────────────────────┬──────────────────────────────────►
  │                ┌─GENERATION─┐            │
  └─remotebind=────┴────────────┴──RUNTIME───┘

►─┬──────────────────────────────────────┬─┬─────────────────────────────┬──────►
  └─contable=─┬─conversion table name──┬─┘ └─location=─┬─EZELOC──────┬──┘
              ├─'*'────────────────────┤               └─system name─┘
              ├─EZECONVT───────────────┤
              ├─NONE───────────────────┤
              └─BINARY─────────────────┘

►─┬───────────────────────────┬─┬─────────────────────────────┬─────────────────►
  │              ┌─CLIENT─┐   │ └─remoteapptype=─┬─VG────┬────┘
  └─luwcontrol=──┴─SERVER─┴───┘                  ├─NONVG─┤
                                                 └─ITF───┘

►─┬─────────────────────────────────────┬──────────────────────────────────────►◄
  └─serverid=──server identifier────────┘
```

The following attributes are ignored:

- externalname
- library
- remotecomtype

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA location=CICS1
     serverid=CS07 contable=ELACNENU LUWCONTROL=CLIENT
```

Configuring a CICS for Windows NT Server

Summary Table of Valid Clients and Protocols

Table 34. Valid Clients and Protocols for VisualAge Generator Servers on the CICS for Windows NT Platform

Server Platform	**Client Platforms**					
	OS/2 (GUI, C++, ITF)	**Windows 95 and Windows 98 (GUI)**	**Windows NT (GUI, C++, ITF)**	**AIX (C++)**	**CICS**	**Solaris (C++)**
CICS for Windows NT	CICS Client	CICS Client	CICS Client	CICS Client	CICS DPL	CICS Client
Notes: CICS as a Client Platform refers to: – CICS for AIX – CICS for MVS/ESA – CICS for OS/2 – CICS for Windows NT – CICS for VSE/ESA – CICS for Solaris						

CICS Client Protocol

List of Valid Clients

- OS/2 (GUI, C++, ITF)
- Windows 95 and Windows 98 (GUI)
- Windows NT (GUI, C++, ITF)
- AIX (C++)
- Solaris (C++)

Identifying the Server Location

Specify the server location (system identifier) in the LOCATION linkage table attribute. You can also set the location dynamically in the client program at run time using EZELOC. If the server location is not specified, the default is the first entry in the CICS Client initialization file.

Linkage Table Attributes for Generating CICS Server Programs

Specify the following calllink attributes when you generate CICS server programs:

```
►►──:calllink──applname=program name──linktype=─REMOTE──┬─────────────────────┬──────►◄
                                                         └─parmform=─COMMDATA──┘
```

The following attributes are ignored:

- contable

- externalname
- library
- location
- luwcontrol
- remoteapptype
- remotebind
- remotecomtype
- serverid

Example Linkage Table

```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA
```

Considerations for Defining the Server Program

If the remote program performs printing, it must move the print destination into EZEDESTP.

Server Program Set Up

The server program is generated and prepared as is any other CICS program on the server system. The server transaction must be defined to CICS on the server system. The default transaction name associated with a server program is the CICS-supplied mirror transaction, CPMI.

Specifying Parameter Format

Parameter format (the `parmform` attribute on the calllink tag) can be specified as COMMDATA (data values passed in the COMMAREA) or COMMPTR (pointers passed in the COMMAREA).

Using COMMPTR Format: Use COMMPTR for calling COBOL subroutines that were developed to use the non-VAGen parameter passing convention that uses pointers in the CICS COMMAREA. When the pointer format is used, the generated client program calls an intermediate catcher program (ELACSV, provided with VisualAge Generator host services) via the ECI instead of calling the server program directly. The catcher program receives the parameter data from the client, builds a pointer list in the COMMAREA that points to the individual parameters, and links to the server program. A program processing table (PPT) must be defined for the ELACSV program on the CICS server system.

COMMPTR format is only supported for MVS CICS, CICS for VSE/ESA, and CICS OS/2 servers.

Using COMMDATA Format: Use COMMDATA for calling generated programs and for calling COBOL subroutines where possible. COMMDATA provides better performance in calling remote programs. With COMMDATA, the server program is called directly without going through the catcher program.

CICS DPL Protocol

List of Valid Clients
- CICS for AIX
- CICS for MVS/ESA
- CICS for OS/2
- CICS for Windows NT
- CICS for VSE/ESA
- CICS for Solaris

Identifying the Server Location
You can specify the server location (CICS system identifier) on the `location` linkage table attribute, or dynamically set the location in the client program at runtime using EZELOC. If the location is not specified, the default location is the system identifier associated with the server program in the CICS program definition on the client system.

Linkage Table Attributes for Generating CICS Server Programs
Specify the following calllink attributes when you generate CICS server programs:

```
►►──:calllink──applname=program name──linktype=─REMOTE──┬─────────────────────┬──────►◄
                                                        └─parmform=─COMMDATA──┘
```

The following attributes are ignored:
- contable
- externalname
- library
- location
- luwcontrol
- remoteapptype
- remotebind
- remotecomtype
- serverid

Example Linkage Table
```
:calllink applname=ELACVP5 linktype=REMOTE parmform=COMMDATA
```

Considerations for Defining the Server Program
If the remote program performs printing, it must move the print destination into EZEDESTP.

Server Program Set Up

The server program is generated and prepared as is any other CICS program on the server system. In addition, there must be a CICS program definition for the server program on the CICS client system containing the following information:

- Resident option = remote
- Remote system identifier
- Remote program name (optional)
- Remote transaction identifier (optional)

The server transaction must be defined to CICS on the server system. The default transaction name associated with a server program is the CICS-supplied mirror transaction, CPMI.

Part 3. Java Wrappers and Enterprise Beans

Chapter 18. VisualAge Generator JavaBeans Wrappers and Enterprise Beans

VisualAge Generator generates JavaBeans wrapper classes for calling server (remote called batch) programs via VisualAge Generator PowerServer middleware.

The generated classes represent the following:
- Server programs
- Record parameters
- Rows in substructured record arrays

The JavaBeans wrappers perform the following functions on the server call for the developers:
- Data marshalling and format conversion on server calls between the following:
  - Objects and record structures
  - Unicode and ASCII or EBCDIC code pages
  - Floating point and decimal or packed decimal numbers
- Extended unit of work control for calls to CICS or OS/400 servers.

The classes can be used by Java developers in building Java applications, applets, servlets, and JavaServer Pages (JSP). When used in Java applications, the server wrapper calls the VisualAge Generator PowerServer API from the system on which the application is running. When used in Java servlets, and JSPs, the server wrapper calls the VisualAge Generator PowerServer API from the system on which a web application server starts the servlet or JSP.

When used in Java applets, the wrapper runs within a Java-enabled browser and uses Java Remote Method Invocation (RMI) to request that the PowerServer API be called from a VisualAge Generator Java gateway. You must start a CSOSessionManager class on the VisualAge Generator Java gateway to listen for RMI requests. See "How to Start the Session Manager on your VisualAge Generator Java gateway" on page 338 for more details.

VisualAge Generator run-time support includes Java classes for server, record, and record array superclasses, plus classes for managing unit of work and PowerServer linkage control. The Java support is shipped with the OS/2, Windows, AIX, MVS, and AS/400 run-time products, supporting Java applications and web servers running on these systems.

VisualAge Generator Java Package

A Java package named ″com.ibm.vgj.cso″ is shipped with VisualAge Generator Common Services for OS/2, Windows, and AIX. The package is also shipped with VisualAge Generator Server for MVS. The classes in the package are used with the generated beans to communicate with VisualAge Generator server programs.

Supplemental Documentation

In case you need to generate Java wrappers for your server programs and send the output to Java developers who do not have VisualAge Generator Developer on Java or VisualAge Generator Developer on Smalltalk, HTML formatted documentation on how to use Java wrappers is provided in a file shipped with VisualAge Generator Common Services. This documentation also contains sample output from generating the server program STAFFMN. See figure .... for the ESF specification for the STAFFMN program.

Documentation on the V4.5 Java Wrapper support is included in the Common Services component of the product for Windows NT, OS/2 and AIX. The documentation is shipped as a dataset containing compressed HTML files. The name of this file and the mechanism for uncompressing it is dependent on the system on which the Common Services component is installed.

Uncompressing the Supplemental Documentation File on Windows NT

On Windows NT the documentation is contained in a self-extracting file named javawin.exe. This file is placed in the exe directory relative to the root directory where Common Services is installed. The default for this directory is C:\IBMVAGEN\VGCSO45\EXE. To uncompress the HTML files:

1. Create a directory, such as c:\wrapdoc, and change to that directory.
2. Run the executable javawin.exe from the c:\wrapdoc directory to expand the html files into c:\wrapdoc. (After installing the Common Services component, javawin.exe should be in your path.)

Uncompressing the Supplemental Documentation File on OS/2

On OS/2 the documentation is contained in a self-extracting file named javaos2.exe. This file is placed in the exe directory relative to the root directory where Common Services is installed. The default for this directory is C:\VGCSO45. To uncompress the HTML files:

1. Create a directory, such as c:\wrapdoc, and change to that directory.
2. Run the executable javaos2.exe from the c:\wrapdoc directory to expand the HTML files into c:\wrapdoc. (After installing the Common Services component, this executable should be in your path).

Uncompressing the Supplemental Documentation File on AIX

On AIX the documentation is contained in tar file csojavadoc.tar. This file is installed in the root directory where VisualAge Generator Server on AIX is installed. The default for this directory is /usr/lpp/vgwgs45. To uncompress the HTML files:

1. Create a directory, such as /home/username/wrapdoc, and change to that directory.
2. From the new directory, run the command tar -xvf /usr/lpp/vgwgs45/csojavadoc.tar to expand the HTML documentation files into the new directory.

After expanding the compressed documentation file, use a web browser to browse file csojava.html. For example, on Windows NT specify a URL of file:///c:/wrapdoc/csojava.html.

Installing the com.ibm.vgj.cso Package for V4.5

For VisualAge Generator Common Services V4.5 or later, for VisualAge Generator Server for MVS V1.2 or later, and for VisualAge Generator Server for AS/400 V4R5 or later, the com.ibm.vgj.cso class files are compressed into file hpt.jar. Except for OS/390 and AS/400 platforms, this file is placed in the directory where VisualAge Generator Common Services V4.5 is installed. If you install the Java support for OS/390 as part of the installation of VisualAge Generator Server for MVS, hpt.jar is placed in the directory specified for that optional install.

For VisualAge Generator Server for AS/400, in addition to hpt.jar, you need to include jt400.jar in CLASSPATH for a non-GUI Java application or when you are starting the CSO Session Manager. Jt400.jar is located in the /QIBM/ProdData/HTTP/Public/jt400/lib directory.

To make the com.ibm.vgj.cso package accessible to Java, make sure the Java CLASSPATH environment variable where the Common Services code runs includes the hpt.jar file. For example, if you installed Common Services in the directory C:\HPTCSOW on a Windows machine, then specify:

```
 set CLASSPATH=...;C:\IBMVAGEN\VGCSO45\hpt.jar;...
```

For AIX, if the Java support was installed in directory /usr/lpp/vgwgs45, then specify:

```
export CLASSPATH=...:/usr/lpp/vgwgs45/hpt.jar:...
```

For OS/390, if the Java support was installed in directory /usr/lpp/vgwgs45, then specify the following:

```
export CLASSPATH=...:/usr/lpp/vgwgs/hpt.jar:
```

For AS/400, if environment variable CLASSPATH is not set, run the following command:

```
ADDENVVAR ENVVAR('CLASSPATH') VALUE('/QIBM/ProdData/HTTP/Public/jt400/lib/jt400.jar:/QVGEN/LIB/hp
```

If CLASSPATH is set, run the following command:

```
CHGENVVAR ENVVAR('CLASSPATH') VALUE('...:/QVGEN/LIB/hpt.jar')
```

where ... is the previous value of CLASSPATH.

com.ibm.vgj.cso Package Classes

The com.ibm.vgj.cso package includes the following class definitions:

com.ibm.vgj.cso.CSOPowerServer
CSOPowerServer is the interface implemented by all classes providing support for calling the Power Server API, whether locally or remotely. Classes implementing this interface establish a communication session with the VisualAge Generator PowerServer API for the purpose of calling VisualAge Generator server programs as well as committing or rolling back extended units of work.

The class is intended for use only by other classes of the com.ibm.vgj.cso package.

com.ibm.vgj.cso.CSORemotePowerServerProxy
CSORemotePowerServerProxy establishes a communication session between a Java applet and the VisualAge Generator PowerServer API.

See com.ibm.vgj.cso.CSORemotePowerServerProxy.html in the package directory where you installed the supplemental documentation for class API documentation. See “Supplemental Documentation” on page 290 for more information.

com.ibm.vgj.cso.CSOLocalPowerServerProxy
CSOLocalPowerServerProxy establishes a communication session between a Java application, servlet, or JSP and the VisualAge Generator PowerServer API.

See com.ibm.vgj.cso.CSOLocalPowerServerProxy.html in the package directory where you installed the supplemental documentation for class API documentation. See “Supplemental Documentation” on page 290 for more information.

com.ibm.vgj.cso.CSORemotePowerServerImpl
CSORemotePowerServerImpl is a Java server application running on your VisualAge Generator Java gateway that acts as a gateway to the PowerServer API for applets. This Java application is started when you start the CSOSessionManager class on the VisualAge Generator Java gateway. The CSOSessionManager class registers the CSORemotePowerServerImpl class with the RMI Registry. See “How

to Start the Session Manager on your VisualAge Generator Java gateway" on page 338 for more details.

The methods of this class are not used directly by Java clients using generated JavaBeans wrappers.

com.ibm.vgj.cso.CSOException

CSOException is an exception class thrown by wrapper classes when any kind of error occured when accessing the Power Server API.

See com.ibm.vgj.cso.CSOException.html in the package directory where you installed the supplemental documentation for class API documentation. See "Supplemental Documentation" on page 290 for more information.

com.ibm.vgj.cso.CSOCallOptions

CSOCallOptions object contains linkage attributes used to control the PowerServer API. An instance of this class is contained in the *callOptions* variable of each generated server program wrapper.

If REMOTEBIND=GENERATION is specified in the generation-time linkage table, information from the linkage table is stored in the *callOptions* variable. No linkage table is read at run time.

If REMOTEBIND=RUNTIME is specified in the generation-time linkage table, only the linkage table name is stored in the *callOptions* variable. Linkage parameters used to locate and call the server program are determined by the linkage table found at run time. On a VisualAge Generator Java gateway, the linkage table is read only once to find the linkage parameters for a server program unless the CSOSessionManager is restarted on the gateway. Subsequent calls to the server program, regardless of client, use the linkage parameters previously found.

The *callOptions* variable may also be used to set the user ID and password to be passed to the Power Server API.

Java clients using generated JavaBeans wrappers may use methods to get and set user ID and password - *getUserId()*, *setUserId(String)*, *getPassword()*, and *setPassword(String)*. These are the only methods of the CSOCallOptions class that should be used directly by Java clients.

This user ID and password is only used to access server platforms like CICS and IMS. It is not used for database access.

See com.ibm.vgj.cso.CSOCallOptions.html in the package directory where you installed the supplemental documentation for class API documentation. See "Supplemental Documentation" on page 290 for more information.

com.ibm.vgj.cso.CSOServerProgram
CSOServerProgram is the super class for generated server wrappers. Its methods are intended only for use by generated beans to marshal parameter data from and back to Java objects when the server program is called.

com.ibm.vgj.cso.CSORecord
CSORecord is the super class for generated record wrappers. A record wrapper class is generated for each record parameter in a server program's called parameter list. Marshalling methods in the class should be used only by generated beans.

com.ibm.vgj.cso.CSORecordArrayRow
CSORecordArrayRow is the super class for wrappers generated to represent a single row in a multiply occurring substructure in a record parameter. Marshalling methods in the class should be used only by generated beans.

com.ibm.vgj.cso.CSOEZEDLPSB
CSOEZEDLPSB is a subclass of Record, and is used in a generated server wrapper for a server program that accesses an IMS database. A server program that accesses an IMS database must be passed the EZE word, EZEDLPSB, which contains the name of the PSB to be used when accessing the IMS database, as well as some internal data. The CSOEZEDLPSB class correctly passes all data required for EZEDLPSB functionality, and provides methods to get and set the PSB name.

See com.ibm.vgj.cso.CSOEZEDLPSB.html in the package directory where you installed the supplemental documentation for class API documentation. See "Supplemental Documentation" on page 290 for more information.

com.ibm.vgj.cso.UnitOfWork
UnitOfWork is an interface implemented by AppletUnitOfWork and ApplicationUnitOfWork. UnitOfWork establishes a communication session with the VisualAge Generator PowerServer API for the purpose of calling VisualAge Generator server programs plus committing or rolling back extended units of work. This class was deprecated in V4.5, and the CSOPowerServer interface should be used instead. The interface may be deleted in a future release.

See com.ibm.vgj.cso.UnitOfWork.html in the package directory where you installed the supplemental documentation for class API documentation. See "Supplemental Documentation" on page 290 for more information.

com.ibm.vgj.cso.AppletUnitOfWork
AppletUnitOfWork establishes a communication session between a

Java applet and the VisualAge Generator PowerServer API. This class was deprecated in V4.5, and the CSORemotePowerServerProxy class should be used instead.

See com.ibm.vgj.cso.AppletUnitOfWork.html in the package directory where you installed the supplemental documentation for class API documentation. See "Supplemental Documentation" on page 290 for more information.

com.ibm.vgj.cso.ApplicationUnitOfWork
ApplicationUnitOfWork establishes a communication session between a Java application, servlet, or JSP and the VisualAge Generator PowerServer API. This class was deprecated in V4.5, and the CSOLocalPowerServerProxy class should be used instead.

See com.ibm.vgj.cso.ApplicationUnitOfWork.html in the package directory where you installed the supplemental documentation for class API documentation. See "Supplemental Documentation" on page 290 for more information.

Migrating Java Clients from Previous Releases

The classes used to access server programs from a Java client were enhanced in VisualAge Generator version 4.0 so that they could be used by both generated wrappers and by VisualAge Generator GUIs developed in the VisualAge for Java visual composition editor. Some new capabilities were also added for record wrapper generation. As a result, if you have developed Java clients using generated JavaBeans wrappers in V3.0 or V3.1 of VisualAge Generator, there are a few steps you must take to migrate those Java clients to use the JavaBeans wrapper support in V4.5. No migration is necessary when you move from version 4.0 to version 4.5.

Regenerate JavaBeans Wrappers

Any JavaBeans wrappers that you use in your Java clients to call server programs must be regenerated using VisualAge Generator Developer on Java. The new JavaBeans wrappers will then access the new V4.5 run-time classes used to call a server program.

Change References to the ibm.cso Package

In order to meet standard naming conventions for Java packages, the name of the package containing JavaBeans wrapper support classes was changed from ibm.cso to com.ibm.vgj.cso. As a result, you must change this package name whenever you imported the ibm.cso package or you fully qualified the class type when you declared a variable using one of the classes of the ibm.cso package. If you regenerate the JavaBeans wrappers used by your Java clients, the ibm.cso package references from within the wrappers will reference the new package name. However, explicit references to the ibm.cso package by your Java clients must be changed ″manually″.

It is very likely that the only ibm.cso classes that were referenced explicitly by your Java clients are the ibm.cso.UnitOfWork, ibm.cso.ApplicationUnitOfWork, and ibm.cso.AppletUnitOfWork classes. You can use the features of VisualAge for Java to find references to these classes.

Deprecated Classes

While the UnitOfWork classes continue to function as they did in previous releases, they have been deprecated, and may be removed in a future release. The new classes that provide equivalent function are:

Table 35. Deprecated and replacement classes

Deprecated Class	Replacement Class
ibm.cso.UnitOfWork	com.ibm.vgj.cso.CSOPowerServer
ibm.cso.ApplicationUnitOfWork	com.ibm.vgj.cso.CSOLocalPowerServerProxy
ibm.cso.AppletUnitOfWork	com.ibm.vgj.cso.CSORemotePowerServerProxy
ibm.cso.UnitOfWorkServerImpl	com.ibm.vgj.cso.CSORemotePowerServerImpl

If your Java client is an applet, you used the AppletUnitOfWork(Applet) or the AppletUnitOfWork(Applet, String) constructor to create an instance of the AppletUnitOfWork class, and you want to change to use the CSORemotePowerServerProxy class, you can no longer just pass your applet to the CSORemotePowerServerProxy constructor. Instead use the CSORemotePowerServerProxy(String) constructor as follows:

```
CSORemotePowerServerProxy(this.getCodeBase().getHost());
```

where this is a reference to your applet. See "How to Call Server Wrappers from Applets" on page 335 for a complete example.

Conversion in Java Virtual Machine

The conversion features of the Java run-time library are used to perform the conversion between Java Unicode and the server program code page. The conversion table names specified in the linkage table are used to determine the correct Java run-time conversion to use, so the contable attribute is required in the linkage table.

There is a change in the conversion table name for English and Portuguese EBCDIC hosts due to the Java run-time conversion. All uses of the CSOE37 conversion table in linkage tables should be changed to CSOE037.

Specifying What to Load into Your Image for Generation

When using batch generation in previous versions of VisualAge Generator, the /CONFIGMAPNAME and /CONFIGMAPVERSION generation options were used to indicate what had to be loaded into the image to be sure all referenced classes are available. JavaBeans wrapper generation is available only in VisualAge Generator Developer on Java. For generation in VisualAge Generator Developer on Java, you need to specify the /PROJECT generation option instead. See the VisualAge Generator Generation Guide for details on invoking batch generation.

Specifying Package Name for Generated JavaBeans Wrappers

In previous releases the package name given to generated JavaBeans wrappers was set to the server name appended with the character ″P″. This forced you to place wrappers generated for each server program into their own package. When the same record parameter was used by several programs, you either had to keep multiple copies of the record wrapper, or you had to modify generated wrappers to change package names. The restriction on package name has been removed. You must now specify the /PACKAGENAME generation option when generating JavaBeans wrappers to specify the package where the wrappers are to be placed. If you can place all generated wrappers for a subsystem into the same package, you can avoid keeping multiple copies of record wrappers and modifying wrappers to change package names.

Starting a Remote Unit of Work for Applets

In previous releases you started a Java application which registered the ibm.cso.UnitOfWorkServerImpl class with the Remote Method Invocation registry in order to respond to Power Server requests made by an applet. You could specify some run-time arguments when starting the application. The ibm.cso.UnitOfWorkServerImpl class started a Session Manager that allowed you to perform adminstrative functions to monitor applet run-time activity.

In V4.5, you start the com.ibm.vgj.cso.CSOSessionManager class with run-time arguments specified using a different syntax. The Session Manager registers the com.ibm.vgj.cso.CSORemotePowerServerImpl class with the Remote Method Invocation registry to respond to Power Server requests made by applets. See “How to Start the Session Manager on your VisualAge Generator Java gateway” on page 338 for details on starting the Session Manager and on specifying run-time arguments.

Enhancements in VisualAge Generator Developer on Java V4.5

In V4.5 several enhancements were made to improve wrapper generation. You may want to make changes to your Java clients to take advantage of these enhancements.

- Filler data items defined in record parameters are now allowed. Java variables are generated into the record and record array row wrappers with generated names. No access methods are generated for the filler data items, but data in their positions is maintained (sent/returned) on calls to server programs.
- All record types may now be passed as parameters. If a record parameter is an SQL record, additional get/set methods are generated for data items to

allow you to query and set their null indicators. See the VisualAge Generator Generation Guide for details.

Specifying a User ID and Password

You can pass a user ID and password from your Java client using the setUserId and setPassWord methods of the CSOCallOptions class. If you do this, the Java client must prompt for the user ID and password values. Each server program wrapper contains an instance of the CSOCallOptions class in variable callOptions. To set user ID and password for server program wrapper instance staffmn, for example, code the following:

```
staffmn.callOptions.setUserId("USERID") ;
staffmn.callOptions.setPassWord("PASSWORD") ;
```

This user ID and this password are only used to access server platforms like CICS and IMS. They are not used for database access.

You can also pass a user ID and password from your Java client using the HptCommSession part. This part contains two additional properties, password and userID, which can be set statically in the program or during program execution. To use this method, first drop a VAGenCommSession part onto your Java free-form surface. If you want to set this value automatically in the code, you need to update the password and userID properties in the HptCommSession part. If you want to receive the value during run time, take the following actions:

1. Code a prompt for the user to enter those values.
2. Connect these values to the HptCommSession part properties, user ID to user ID property, and password to password property.
3. Connect the VAGenCommSession part (this) to the VAGenCommSession of the bean.

For more information on using VAGenCommSession part, refer to User’s Guide.

Calling Server Programs as Session Beans

Beginning with V4.0 of VisualAge Generator for Java you were able to generate a session bean that calls a COBOL or C++ server program. The session bean support available in VisualAge Generator prior to V4.5 allowed you to access a server program through a session bean deployed on an Enterprise Java Server (EJS). However, the server program did not do two things:

- Run in the same EJS process as the session bean that called the server program. The server program was called through VisualAge Generator

Common Services middleware, and did not even have to be executed on the same machine as the session bean.

- Obtain its database connections from an EJS container. Thus, database updates could not be coordinated by the EJS transaction services.

As a result, server programs called through session beans prior to V4.5 could not truly participate in EJS transactions. Each call of a server program started a new EJS transaction, but any database updates were managed by CICS, IMS, or database manager transactions as far as commits and rollbacks were concerned.

The previous capability for calling a server program as a session bean is still available, but with the addition of Java server generation in V4.5, a server program generated in the Java language can fully participate in EJS transactions.

Requirements for Generating Session Beans that Participate in EJS Transactions

In order for a server program to participate in an EJS transaction, the following things must be true.

- The server program must be generated in the Java language rather than COBOL or C++.
- Either:
  - A session bean must be generated to invoke the server program in an EJS process.

    In this case the EJBGROUP generation option must be specified when generating the session bean, and the server program’s linkage table entry in the linkage table used to generate the session bean must specify the following attributes.

    - LINKTYPE=SESSIONEJB
    - REMOTEAPPTYPE=VGJAVA
    - REMOTECOMTYPE=DIRECT

    In addition, if session bean A (as opposed to a servlet) invokes server program B as a session bean, then the linkage table entry for program B within the linkage table used to generate server program A must include the following specifications.

    - LINKTYPE=SESSIONEJB
    - REMOTEAPPTYPE=VGJAVA
    - REMOTECOMTYPE=DIRECT

    See “Generation/Runtime Setup Examples” on page 306 for more information on linkage table attributes.

  – The server program is not generated as a session bean, but is one of a chain of called Java server programs (not called as session beans) where the first program of the chain is called from a Java server program session bean that participates in an EJS transaction. Each program of the call chain must be invoked with the linkage attribute LINKTYPE=DYNAMIC or LINKTYPE=STATIC
- The only I/O to be controlled by the EJS transaction are updates to SQL databases. File I/O is not controlled by EJS transactions. See “Generation/Runtime Setup Examples” on page 306 for more information on linkage table attributes when using session beans.

EJS Transaction Demarcation for Generated Session Beans

The session beans generated by VisualAge Generator do not perform their own transaction demarcation. (They do not support a transaction attribute of TX_BEAN_MANAGED in EJB terminology.) Instead, they depend on either the container in which they are deployed, or the client invoking them, to start any EJS transaction in which they are to participate. This has implications for how you must design server programs in order to control when commits and rollbacks are triggered.

Because a generated session bean does not start or end its own transactions, commits and rollbacks can only be performed outside the boundary of a call to the session bean. Thus, commits and rollbacks of database updates made by generated session beans are performed only at the return from a call. If the client invoking the session bean starts the EJS transaction in which the session bean is running, then the client controls when the transaction is ended. If the transaction is started by an EJS container’s invocation of a session bean method, then the container ends the transaction when the method ends. (For this discussion, throwing an exception due to a system problem is one way in which a method may be ended.) In either case the session bean can force a rollback to occur when the transaction is ended.

This means that calls to EZECOMIT cannot force an intermediate commit of updates during a call. Calls to EZEROLLB simply mark the current transaction for rollback. Once a call to EZEROLLB is made, there is no use in performing additional I/O to relational databases because those updates, along with any previous updates made during the call, will be rolled back. A call to EZECOMIT from within a generated session bean, or from a chain of called Java server programs running in the same process as a generated session bean, is essentially a no-op. It does not cause a commit of previous updates, and it does not cause any cursor resetting. Similarly, the only function performed by EZEROLLB is to mark the current EJS transaction to eventually force a rollback when the transaction ends.

Because commits and rollbacks can only be performed at call boundaries, it is necessary to structure programs so that only database updates that belong to one transaction can be performed within a call. In order to make an update to a database and have that update committed regardless of what happens in further processing, you must exit the server program. You can make updates to more than one database within a call, but all updates made within a call are either committed or rolled back together, regardless of the database in which the updates were made.

EJS Container Usage of a Session Bean's Transaction Attribute for Transaction Demarcation

One of the deployment attributes that you can specify in a session bean's deployment descriptor is the transaction attribute. This attribute is used to specify whether or not the methods of the session bean are to participate in an EJS transaction and, if so, whether an existing transaction is to be inherited or a new transaction is to be started when a method is invoked. The following values may be specified for generated session beans:

TX_REQUIRED
Specifies that if a current transaction context exists, methods are invoked in that transaction context. If no transaction context exists, the container starts a new transaction before making a method call and attempts to commit the transaction when the method has completed. The container performs the commit protocol before the method result is sent to the invoker. If the session bean invokes a method of another session bean, the invoking session bean's transaction context is passed to the invoked session bean.

This is the default transaction attribute value for generated session beans.

TX_REQUIRES_NEW
Specifies that the container always starts a new transaction before making a method call and attempts to commit the transaction when the method has completed. If there is an existing transaction context, that transaction context is suspended before the new transaction is started, and resumed when the new transaction has completed. The container performs the commit protocol before the method result is sent to the invoker. If the session bean invokes a method of another session bean, the invoking session bean's transaction context is passed to the invoked session bean.

TX_MANDATORY
Specifies that the session bean must always be called with an existing transaction context. Otherwise, the container throws the TransactionRequired exception. The container does not perform any commit processing before returning. If the session bean invokes a method of another session bean, the invoking session bean's transaction context is passed to the invoked session bean.

TX_SUPPORTS
Specifies that the session bean uses an existing transaction context, if any. If there is no existing transaction context, methods are invoked without transaction scope. If the session bean invokes a method of another session bean, the invoking session bean's transaction context, if any, is passed to the invoked session bean.

TX_NOT_SUPPORTED
Specifies that session bean methods are invoked without a transaction scope. If there is an existing transaction context, that transaction is suspended before the container invokes a method and is resumed on completion of the method. If the session bean invokes a method of another session bean, the invoking session bean's transaction context, if any, is not passed to the invoked session bean.

Using the Transaction Attribute to Control Transaction Demarcation

Previous sections discussed how you must structure your server programs so that no EZECOMIT calls are necessary within the server program to control commit processing. This section discusses how to specify transaction attributes so that transaction contexts extend across a chain of server program calls.

Suppose you are designing a simple order/entry application where one server program performs all updates to an ORDER table and a second server program performs all updates to an ORDER_ITEM table. When an order is received, the order has to be processed so that both the ORDER table is updated with the new order, and the ORDER_ITEM table is updated with all line items of the order. The ORDER table update may not be committed without the ORDER_ITEM table updates being committed, and vice versa.

To force this behavior you can define a third program that controls processing of the order. It receives all the data associated with the order, calls the program that makes the ORDER table update, then calls the program that makes the ORDER_ITEM table updates. The control program would be deployed with a transaction attribute of TX_REQUIRES_NEW. The ORDER and the ORDER_ITEM programs would be deployed with a transaction attribute of TX_REQUIRED. Then the processing of both the ORDER and the ORDER_ITEM program calls would be performed within the transaction context of the control program. When the call to the control program completes, the updates would be committed or rolled back depending on whether any of the three programs called EZEROLLB, and whether an exception was thrown due to some system problem. It does not matter whether the control program makes one call to the ORDER program and one call to the ORDER_ITEM program, or whether the control program makes one call to the ORDER program and a call to the ORDER_ITEM program for each item of the order. The transaction does not end until the control program ends.

Alternatively, the function of the control program could be moved back to the client servlet. The servlet can start an EJS transaction, then call the session bean that updates the ORDER table, followed by a call to the session bean that updates the ORDER_ITEM table. The servlet would have to issue a commit or rollback based on return information from the two session beans to end the transaction. You might deploy the ORDER and ORDER_ITEM session beans with a transaction attribute of TX_MANDATORY if they would always be invoked by a client servlet that starts the EJS transaction. This servlet-controlled transaction design would not usually be the preferred design because:

- Two calls across the network are required instead of one
- Code has to be developed in the client servlet to manage an EJS transaction. The EJS container already provides the necessary function.

However, there are situations where it is necessary to have EJS transactions controlled from the client. See section ″Implementing Client Controlled EJS Transactions″ for details on how servlets can start their own EJS transactions, called user transactions in the EJB specification.

Now let’s change the requirements for the programs processing orders. Suppose an order is for a new customer. Then we want the customer information to be added to a CUSTOMER table, regardless of whether the order can be processed successfully or not. A session bean can be written to handle updates/inserts to the CUSTOMER table, and given a transaction attribute of TX_REQUIRES_NEW. The control program can be changed to first call the CUSTOMER session bean to handle customer information. When this call is made a new transaction will be started by the EJS container before invoking the CUSTOMER session bean’s call method, and the transaction will be completed by the container on return from the call method. While the CUSTOMER session bean is processing, the transaction context of the control program is suspended. The transaction context of the control program is resumed on return from the CUSTOMER session bean. If CUSTOMER processing is successful, the control program calls the ORDER session bean and the ORDER_ITEM session bean as above. The updates made during these calls are committed or rolled back together when the control program ends.

Implementing Client Controlled EJS Transactions

Figure 13 on page 304 provides code snippets of an EJB client creating a reference to a UserTransaction object and then using that object to begin a transaction and attempt to commit the transaction. Consult WebSphere documentation, for example, ″Writing Enterprise Beans in Websphere″, for more detail on User Transactions.

```
...
import javax.transaction.*;
import javax.naming.*;
...
// Use JNDI name jta/usertransaction to locate the UserTransaction object
Context initialContext = new InitialContext();
UserTransaction tranContext = (UserTransaction)initialContext.lookup("jta/usertransaction");
// Set the transaction timeout to 30 seconds
tranContext.setTransactionTimeout(30);
...
// Begin a transaction
tranContext.begin();
...
// Prepare parameters and call session bean
...
// Try to commit the transaction. If the session bean called EZEROLLB,
// The commit will fail and a rollback will be performed.
// An alternative is for the session been to set a return value, and
// based on that return value, the client can invoke either commit() or rollback()
tranContext.commit();
```

Figure 13. Managing Transactions in an EJB Client

VisualAge Generator Runtime Usage of Linkage Attributes

If a session bean is to call a server program that is not a session bean, and the called server program is to participate in the session bean's EJS transaction, then the session bean's generation linkage table entry for the called server program must specify the following:

- LINKTYPE=DYNAMIC or LINKTYPE=STATIC
- REMOTEAPPTYPE=VAJAVA
- PACKAGENAME='<server program package name>'

In this case, no linkage parameters are searched for at run time.

Otherwise, no database updates made by the server program will be controlled by the session bean's EJS transaction. If the linkage table used at generation time specifies REMOTEBIND=GENERATION for a called server program's entry, and a valid REMOTECOMTYPE is specified, the generation linkage table's linkage attributes are used at run time. If the linkage table used at generation time specifies REMOTEBIND=RUNTIME and does not specify LINKTYPE=DYNAMIC or LINKTYPE=STATIC for a called server program's entry, you must supply a run-time linkage table or a property file specifying how to call the server program. When the calling program is a Java server program, the VisualAge Generator runtime searches for property files and run-time linkage tables in the following manner:

1. Search for file <prgName>.properties in the CLASSPATH, <where prgName> is the name of the called program. If an entry for the called program is not found,
2. Search for file vgj.properties in the CLASSPATH. If an entry for the called program is not found,

3. If the LINKAGE generation option was specified at generation time,
   a. If the LINKAGE generation option specifies a fully qualified dataset name, that dataset is read. Otherwise,
   b. Search for the specified linkage table in directories in a platform dependent order:

      For Windows platforms
      1) Search the current directory.
      2) Search directories specified by the PATH environment variable.

      For the OS/2 platform
      1) Search the current directory.
      2) Search directories specified by the DPATH environment variable.

      For UNIX-based platforms (including OS/390 USS)
      Search directories specified by the DPATH environment variable.
4. If no LINKAGE generation option was specified, or the linkage table specified was not found in step 3, the CSOLINKTBL environment variable is read. (When the call is being made in an application server environment like WebSphere, the CSOLINKTBL environment must be set so that it is available in that environment. For example, if WebSphere is running as a service on a Windows NT system, the CSOLINKTBL must be set as a system environment rather than a user environment variable. An application server may have its own way of making environment variables available to EJBs.) If a value is specified for CSOLINKTBL:
   a. If the CSOLINKTBL value is a fully qualified dataset name, that dataset is read.
   b. If the CSOLINKTBL value is not a fully qualified dataset name, the specified dataset is searched for in a directory as described in steps 3.a and 3.b.
5. If no linkage table entry is found by either step 3 or step 4, the call will fail.

The VisualAge Generator runtime searches for run-time linkage tables as described in steps 3 through 5 for Java server programs. See the appendix on Java properties files in the *VisualAge Generator Generation Guide* for more detail on specifying linkage parameters within property files, and how property files can be generated from linkage tables. See the *VisualAge Generator Client/Server Communications Guide* for more detail on specifying linkage parameters within linkage table files.

Generation/Runtime Setup Examples

There are many configurations possible for systems containing combinations of servlets, C++ or COBOL server programs, Java server programs, Java server programs as session beans, and C++ or COBOL server programs as session beans. This section describes some possible configurations and the generation options, linkage table attributes, and properties necessary to make them work in a run-time environment. In the following scenarios, the PACKAGE linkage attribute is shown to be ’my.pkg’. This value should be replaced by an actual package name.

Scenario 1

In this scenario a servlet calls COBOL server program PGMA as a session bean. The server program is run in an MVS CICS environment. Because the server program is not a Java server program and is not called by a Java server program, updates made by the server program are not controlled by an EJS transaction. If program PGMA were to call other server programs, database updates by those server programs would also not be controlled by an EJS transaction.

Table 36. Scenario 1

Program	Language	Generation Options	Linkage Table Attribute	EJB?	Transaction Attribute	Updates in an EJS Trans.
PGMA	COBOL	EJBGROUP=MYGROUP	LINKTYPE=SESSIONEJB REMOTECOMTYPE=CICSCLIENT CONTABLE=CSOE047	Y	TX_REQUIRES_NEW (any should work since no updates made in an EJS Transaction)	No

Scenario 2

In this scenario a servlet calls Java server program PGMA as a session bean using a REMOTECOMTYPE of DIRECT. Because the server program is a Java server program that is called using a REMOTECOMTYPE of DIRECT, updates made by the server program are controlled by an EJS transaction. The conversion table CSOJ1252 indicates that the program PGMA uses a Java byte order (the ″J″ in CSOJ1252), and is running on a machine using code page 1252 (English).

Table 37. Scenario 2

Program	Language	Generation Options	Linkage Table Attribute	EJB?	Transaction Attribute	Updates in an EJS Trans.
PGMA	Java	EJBGROUP=MYGROUP	LINKTYPE=SESSIONEJB REMOTECOMTYPE=DIRECT CONTABLE=CSOJ1252 REMOTEAPPTYPE=VGJAVA PACKAGE='my.pkg'	Y	TX_REQUIRES_NEW	Yes

Scenario 3

In this scenario a servlet calls Java server program PGMA as a session bean using a REMOTECOMTYPE of DIRECT. Program PGMA calls Java server program PGMB as a session bean using a REMOTECOMTYPE of DIRECT. Java server program B calls Java server program PGMC as a session bean using a REMOTECOMTYPE of DIRECT. Because all three programs are Java server programs called as session beans using a REMOTECOMTYPE of DIRECT, updates made by any of the three server programs are controlled by an EJS transaction. Because the transaction attribute for session beans PGMB and PGMC is TX_REQUIRED, updates by all three session beans are controlled by the same transaction.

Table 38. Scenario 3

Program	Language	Generation Options	Linkage Table Attribute	EJB?	Transaction Attribute	Updates in an EJS Trans.
PGMA	Java	EJBGROUP=MYGROUP	LINKTYPE=SESSIONEJB REMOTECOMTYPE=DIRECT CONTABLE=CSOJ1252 REMOTEAPPTYPE=VGJAVA PACKAGE='my.pkg'	Y	TX_REQUIRES_NEW	Yes
PGMB	Java	EJBGROUP=MYGROUP	LINKTYPE=SESSIONEJB REMOTECOMTYPE=DIRECT CONTABLE=CSOJ1252 REMOTEAPPTYPE=VGJAVA PACKAGE='my.pkg'	Y	TX_REQUIRES	Yes Inherits from A
PGMC	Java	EJBGROUP=MYGROUP	LINKTYPE=SESSIONEJB REMOTECOMTYPE=DIRECT CONTABLE=CSOJ1252 REMOTEAPPTYPE=VGJAVA PACKAGE='my.pkg'	Y	TX_REQUIRES	Yes Inherits from A

Scenario 4

In this scenario a servlet calls Java server program PGMA as a session bean using a REMOTECOMTYPE of DIRECT. Program PGMA calls Java server program PGMB as a session bean using a REMOTECOMTYPE of DIRECT. Session bean PGMB runs on a different EJS server (at least a different name server resolves its location) so the PROVIDERURL linkage option is specified. Program PGMB runs in its own transaction since transaction attribute TX_REQUIRES_NEW is specified. Java server program PGMA also calls Java server program PGMC as a session bean using a REMOTECOMTYPE of DIRECT. Because program PGMC is a Java server program called as a session bean using a REMOTECOMTYPE of DIRECT, and session bean PGMC has a transaction attribute of TX_REQUIRED, updates made by server programs PGMA and PGMC are controlled by the same EJS transaction.

Note: Even though session bean PGMB's JNDI name is resolved by a different name server than session bean PGMA and may be deployed in a container of a different EJS, if session bean PGMB's transaction attribute were TX_REQUIRED instead of TX_REQUIRES_NEW it would inherit session bean PGMA's transaction. In order for this configuration to be successful, it is necessary for the JDBC driver used by the data sources of all three session beans to support distributed transactions. In J2EE terminology, the JDBC drivers must support the Java Transaction API (JTA). The DB2 V6.1 JDBC driver after FixPack 1 does support JTA. If you need to use other database managers in a distributed transaction environment, you will have to ensure that the databases manager's JDBC driver supports distributed transactions using JTA.

Table 39. Scenario 4

Program	Language	Generation Options	Linkage Table Attribute	EJB?	Transaction Attribute	Updates in an EJS Trans.
PGMA	Java	EJBGROUP=MYGROUP	LINKTYPE=SESSIONEJB REMOTECOMTYPE=DIRECT CONTABLE=CSOJ1252 REMOTEAPPTYPE=VGJAVA PACKAGE='my.pkg'	Y	TX_REQUIRES_NEW	Yes
PGMB	Java	EJBGROUP=MYGROUP	LINKTYPE=SESSIONEJB REMOTECOMTYPE=DIRECT CONTABLE=CSOJ1252 REMOTEAPPTYPE=VGJAVA PROVIDERURL=SERVER2 PACKAGE='my.pkg'	Y	TX_REQUIRES_NEW	Yes In its own transaction
PGMC	Java	EJBGROUP=MYGROUP	LINKTYPE=SESSIONEJB REMOTECOMTYPE=DIRECT CONTABLE=CSOJ1252 REMOTEAPPTYPE=VGJAVA PACKAGE='my.pkg'	Y	TX_REQUIRES	Yes Inherits from A

Scenario 5

In this scenario a servlet calls Java server program PGMA as a session bean using a REMOTECOMTYPE of DIRECT. Program PGMA calls Java server program PGMB, but not as a session bean, using a LINKTYPE of DYNAMIC. Program PGMB calls C++ server program PGMC, but not as a session bean, using a LINKTYPE of DYNAMIC. While server programs PGMB and PGMC are not called as session beans (LINKTYPE is not SESSIONEJB) they do form a chain of called Java server programs with LINKTYPE=DYNAMIC. This causes programs PGMB and PGMC to run in the same process as session bean PGMA. As a result, programs PGMB and PGMC run in the same VisualAge Generator run unit as session bean PGMA, they obtain any database connections from the EJS container, and their updates are controlled by session bean PGMA's transaction. Note that the conversion table specification for dynamic calls is different from the specification for session bean (and remote) calls. For dynamic calls to Java server programs specify CONTABLE=CSOJAVA.

Table 40. Scenario 5

Program	Language	Generation Options	Linkage Table Attribute	EJB?	Transaction Attribute	Updates in an EJS Trans.
PGMA	Java	EJBGROUP=MYGROUP	LINKTYPE=SESSIONEJB REMOTECOMTYPE=DIRECT CONTABLE=CSOJ1252 REMOTEAPPTYPE=VGJAVA PACKAGE='my.pkg'	Y	TX_REQUIRES_NEW	Yes
PGMB	Java		LINKTYPE=DYNAMIC CONTABLE=CSOJAVA REMOTEAPPTYPE=VGJAVA PACKAGE='my.pkg'	N	N/A	Yes Run in A's run unit
PGMC	Java		LINKTYPE=DYNAMIC CONTABLE=CSOJ1252 REMOTEAPPTYPE=VGJAVA PACKAGE='my.pkg'	N	N/A	Yes Run in A's run unit

Scenario 6

This scenario is the same as "Scenario 5" on page 314 except that server program PGMC is a C++ server program called with LINKTYPE = CSOCALL and REMOTECOMMTYPE=TCPIP. The updates for server program PGMB are controlled by session bean PGMA's transaction because server program PGMB runs in session bean PGMA's run unit. Server program PGMC is not a Java server program. It has its own run unit, and its database updates are not controlled by session bean PGMA's transaction.

Table 41. Scenario 6

Program	Language	Generation Options	Linkage Table Attribute	EJB?	Transaction Attribute	Updates in an EJS Trans.
PGMA	Java	EJBGROUP=MYGROUP	LINKTYPE=SESSIONEJB REMOTECOMTYPE=DIRECT CONTABLE=CSOJ1252 REMOTEAPPTYPE=VGJAVA PACKAGE='my.pkg'	Y	TX_REQUIRES_NEW	Yes
PGMB	Java		LINKTYPE=DYNAMIC CONTABLE=CSOJAVA REMOTEAPPTYPE=VGJAVA PACKAGE='my.pkg'	N	N/A	Yes Run in A's run unit
PGMC	C++		LINKTYPE=CSOCALL REMOTECOMTYPE=TCPIP CONTABLE=CSOJ1252 REMOTEAPPTYPE=VG	N	N/A	No

Scenario 7

In this scenario a servlet calls Java server program PGMA as a session bean using a REMOTECOMTYPE of DIRECT. Program PGMA calls Java server program PGMB, but not as a session bean, and using a LINKTYPE of CSOCALL. Because program PGMB is not called with LINKTYPE=SESSIONEJB or LINKTYPE=DYNAMIC, it does not run in session bean PGMA's run unit. Therefore, it does not participate in an EJS transaction.

Table 42. Scenario 7

Program	Language	Generation Options	Linkage Table Attribute	EJB?	Transaction Attribute	Updates in an EJS Trans.
PGMA	Java	EJBGROUP=MYGROUP	LINKTYPE=SESSIONEJB REMOTECOMTYPE=DIRECT CONTABLE=CSOJ1252 REMOTEAPPTYPE=VGJAVA PACKAGE='my.pkg'	Y	TX_REQUIRES_NEW	Yes
PGMB	Java		LINKTYPE=CSOCALL REMOTECOMTYPE=DIRECT CONTABLE=CSOJ1252 REMOTEAPPTYPE=VGJAVA PACKAGE='my.pkg'	N	N/A	No

Configuring Data Sources in WebSphere Advanced Edition V3.5

Session beans generated by VisualAge for Java try to obtain connections from a data source of the EJS container in which the session bean is deployed rather than obtaining the connection directly from a database manager. When you configure a data source within a container, the container can maintain a connection pool so that connections are not automatically released when an EJB indicates it is through with the connection. This makes connecting and disconnecting to databases more efficient. When a connection is obtained from a data source, the container is able to get the information necessary to become the transaction co-ordinator for the connection.

You should reference WebSphere Advanced Edition documentation for detailed information on data sources, connections, and connection pools. The following procedure is provided to help you get started in configuring a data source for use by generated session beans in WebSphere Advanced Edition V3.5. WebSphere may make changes to their Administrative Console user interface, so if you encounter problems refer back to the WebSphere documentation to resolve the problems. The following procedure configures a data source, sampledb, to access the DB/2 sample database.

1. Select View->Type
2. Right-click on JDBCDrivers
3. Select Create
4. On the ″Create a JDBCDriver″ dialog
5. Enter DB2AppDriver in the Name field. (Or choose your own driver name.)
6. From the Implementation class dropdown select COM.ibm.db2.jdbc.app.DB2Driver
7. Leave the URL prefix as jdbc:db2
8. Leave JTA enabled as false unless you have to implement distributed transactions (which are inherently slow). DB2 5.2 does not provided a JTA enabled driver. DB2 6.1 FixPack 1 or later does.
9. Click Create to create the JDBC Driver definition
10. Back on the Type panel, right-click on DataSources and select Create
11. On the ″Create a DataSource″ dialog
12. Enter the data source name sampledb. This will result in a Java Naming and Directory Interface (JNDI) name of jdbc/sampledb being created in the EJS name server. It is this JNDI name that you have to use as the database server name when specifying a default database name for a session bean, or when mapping a symbolic name to a database server name for use in EZECONCT calls. See section ″Configuring the Java Server Runtime to Access EJS Data Sources″ for more detail.

13. Enter sample as the database name. Together with the URL prefix in the JDBC driver configuration, this enables a database URL of jdbc:db2:sample
14. Enter DB2AppDriver (the JDBC driver created above) in the Driver field
15. You can click on the Advanced tab to see what connection pool parameters can be set. See WebSphere documentation on how to use these parameters.
16. Click on Create to create the data source definition

Configuring the Java Server Runtime to Access EJS Data Sources

The VisualAge Generator runtime tries to load property files for a Java server program to gather run-time options for the program. In order for the runtime to find a property file, the file must be included in a directory or in a jar file in the CLASSPATH that is active when the runtime is started. The runtime first looks for a property file named vgj.properties and extracts properties from that property file if found. Then the runtime looks for a property file named <programName>.properties, where programName is the name of the program being invoked. For example, if the program ORDER is being invoked, the runtime will look for file ORDER.properties. (Case is important in property file names.) If a value for a property is specified in <programName>.properties, that value will override the value for the same property specified in vgj.properties.

The following properties may be specified in either the vgj.properties or the <programName>.properties file to provide information used to connect to EJS data sources when a Java server program is invoked as a session bean:

vgj.jdbc.default.database
: Specifies the JNDI name of the EJS data source to be used as the default data source.

  For example, if a data source of sampledb was defined in WebSphere, the JNDI name of the data source is jdbc/sampledb

vgj.jdbc.default.user.id
: Specifies the user ID to use when connecting to the default database

vgj.jdbc.default.user.password
: Specifies the password to use when connecting to the default database

If your session bean uses EZECONCT to connect to and disconnect from a database rather than using a default database, one of the parameters for the EZECONCT call is the name of a database server. When the server program is running as a session bean, the JNDI name of the data source must be specified for this parameter. You can specify the actual JNDI name (for example, jdbc/sampledb) or you can specify a symbolic name that a property maps to a data source JNDI name. If you want to map a symbolic name to a data source JNDI name, specify property

vgj.jdbc.database.<symbolicName> = <dataSourceJNDIName>

For example, if you want EZECONCT to map a symbolic name of SAMPLE to data source sampledb, in your properties file code

vgj.jdbc.database.SAMPLE = jdbc/sampledb

Modifying Deployment Descriptors

After you generate a session bean, you may need to modify its deployment descriptor attributes so that the desired transaction behavior, database locking, and security are applied at deployment.

To edit the deployment descriptor in VA for Java:

- Select the EJB page of the workbench
- Expand the EJB group containing the session bean
- Right-click on the session bean and select Properties
- Select the Bean page of the Properties dialog.

  You can get VisualAge for Java help for this dialog by expanding Help Tasks ″Developing EJB Components″ then ″Setting Descriptor Properties and Generating Deployed Classes″.
- If necessary, select the Transaction Attribute required to get the desired transaction demarcation
- If necessary, select the Isolation Level requried to get the desired database locking.

  See the WebSphere documentation for a description of each of the Isolation Level selections.
- If necessary, select the Run-as Mode to get the desired security checking.

  See the WebSphere documentation for a description of each of the Run-As Mode selections.

Never change the JNDI name for the bean because the VisualAge Generator middleware depends on the generated JNDI name. Also, never change the State Management Attribute from STATEFUL.

Deploying Generated Session Beans

After you have generated a session bean to call a server program, you must use VisualAge for Java to generate the code necessary to deploy the session bean in an enterprise server. New classes are generated that allow client code that you develop to communicate with the session bean using RMI-IIOP, a CORBA interface for distributed object communication. VisualAge for Java provides support for testing your deployed code in a WebSphere test environment. Once you have generated and tested the deployment code, you must export that code and install it in a container of your enterprise server.

Generating Deployed Code for a Generated Session Bean

VisualAge for Java generates deployment code that sends each parameter of the home and remote interfaces of the session bean between the client and the session bean. The new classes generated for a parameter are placed in the same package as the parameter. Because the CSOParameter class of the com.ibm.vgj.cso package is passed on a remote interface method of the generated session beans, VisualAge for Java updates that package when generating deployed code. To allow VisualAge for Java to make updates, you must be sure you have created an open edition of the com.ibm.vgj.cso package (in the IBM VisualAge Generator Runtime project) before generating deployed code for the session bean. Once you have an open edition of the com.ibm.vgj.cso package, to generate deployed code:

1. On the **Enterprise Beans** panel (top left) of the **EJB** page of the Workbench, expand the **EJB Group** into which you generated your session bean
2. Right mouse button click on the generated session bean, and select **Generate** from the context menu, then select **Deployed Code**. The necessary deployed classes are then generated.

Testing a Generated Session Bean

VisualAge for Java can generate a test client class that allows you to test your generated session bean. To run this test client you must manually supply test data for the parameters you need to pass to the server program. If you have simple parameters you may want to take advantage of this feature. For more complex parameters, it is probably more efficient to write your own client. See VisualAge for Java Help for details on generating and calling a test client. You can find the information on testing clients by expanding the ″Tasks″, ″Using the EJB Development Environment″, and ″Testing enterprise beans″ Help topics. These VisualAge for Java Help topics also describe how to start your own client applications in a WebSphere test environment.

Exporting a Session Bean for Deployment

Before you can use your generated session bean in a production environment, you must export the session bean generated by VisualAge Generator and the enterprise server side deployed code generated by VisualAge for Java into a Java archive (JAR) file. You must also export your client code and client side deployed code generated by VisualAge for Java. VisualAge for Java provides facilities for exporting the required classes into server side and client side JAR files. See VisualAge for Java Help for details on exporting code into these JAR files. You can find the information on exporting code by expanding the ″Tasks″, ″Developing EJB Components″, and ″Exporting and Deploying Code″ Help topics.

Exporting Deployed Code for Session Beans: When you export your session bean to a Deployed JAR or an EJB JAR file you will get a SmartGuide that lets you further qualify what classes are to be placed in the JAR file. If you are deploying your generated session beans into WebSphere Advanced Edition, you should click on the ″Select referenced types and resources″ pushbutton to include the classes referenced by the session bean.

You may get warning messages about inner classes (class names with a $ in their name) and about not being able to find the class CSOAS400Driver (if you did not load the IBM Enterprise Toolkit for AS400). You can ignore these warning messages.

Exporting Client Code Using a Session Bean: When you export your client code to a client JAR file you will get a SmartGuide that lets you further qualify what classes are to be placed in the JAR file. By default, VisualAge for Java selects all classes it generated which have to be included in the classpath where any client that uses the session bean is run. If you want to add your client code to this JAR file, instead of creating a separate directory or JAR file that you would add to the classpath used when running your client, you may do so using the following steps:

1. Click on the **Details** push button to the right of the **.class** checkbox in the initial SmartGuide window.
2. In the **Projects** pane of the **.class export** dialog, select the project containing your client code.
3. In the **Types** pane, select the classes required to run the client, including the wrappers used to invoke your session bean.
4. If your client code spans more than one project, you must repeat the previous two steps for each project that includes client code
5. Click on the **OK** push button.

Exporting Java Server Programs Called By a Session Bean

If the session bean you are deploying calls one or more Java server programs, you have to get the class files and property files for the generated Java server programs into a directory or jar file that is included in the session bean container's application server. This may have already been accomplished by sending the generation outputs to the application server machine (using the JAVADESTHOST and JAVADESTDIR generation options) and compiling them there. If you imported the classes into VisualAge for Java in order to test them in the WebSphere Test Environment, you can export them into a directory or jar file.

Because names of generated wrappers for server programs and of Java server programs themselves differ only by case, you will have to place the wrappers and server programs in different jar files or directories on Windows systems.

VisualAge for Java recognizes the differences in the classes, but will not export both a Java server program and its wrapper to the same jar file or directory.

Deploying Session Beans in WebSphere Advanced Edition

Before you are able to run your client code and your session beans, you must deploy the exported JAR files in your application server. The procedure for deploying the exported JAR files depends on the application server you are using. Following is a sequence of steps that you can use to deploy your exported JAR files in WebSphere Advanced Edition:

1. Move your exported client and session bean JAR files to a node where your WebSphere Admin Server is installed.
2. On the WebSphere Advanced Administrative Console **Topology** view, create an application server and a container in which you will deploy your session beans. See the WebSphere Administrator Help for more information. If you are only deploying session beans generated by VisualAge Generator, you can ignore topics concerning entity beans. When you install WebSphere Advanced Edition, an application server named ″Default Server″ with a container called ″Default Container″ is created on an application server node. The Default Server entry also has a web application server called ″servletEngine″ to handle requests for servlets.
3. Select the application server where you want to deploy your session beans and your client code, then select the **General** tab.
4. WebSphere requires that classes defining native methods need to be loaded by the system class loader rather than any of the WebSphere class loaders. Classes in the hpt.jar file of the Common Services component are referenced by generated wrappers and by generated session beans. To cause the system loader to load classes accessing native methods, add a -classpath environment variable to the ″Command line arguments:″ entry field. Include the Common Services hpt.jar file and your client JAR file in the value for the -classpath environment variable. For example, on Windows NT,

   ```
   -classpath C:\IBMVAGEN\VGCSO45\hpt.jar;C:\MyProject\myClient.jar
   ```

   This causes the files or directories specified in the -classpath variable to be prefixed to the classpaths used by WebSphere class loaders.

   If your session bean is calling a Java server you will also need to include in the application server CLASSPATH the Java Server Runtime jar file, vgjwgs.jar and the .class files prepared from the server's generation output, and any property file needed to get linkage attributes to call the Java server. Assuming that the EJB container in which you deploy the session bean is defined in the same application server as the Web Server in which your client servlet was deployed, your classpath specification may be something like:

```
-classpath C:\IBMVAGen\VGCSO45\hpt.jar;C:\IBMVAGen\VGWGS45\vgjwgs.jar;
C:\MyProject\myClient.jar;C:\MyProject\myServer.jar;
```

If your EJB container and your web server are not configured within the same application server, then you will have to place all your client classes in the CLASSPATH of the application server that contains your web server; and place all your Java server classes in the CLASSPATH of the application server that contains the EJB container in which you deploy your session bean. The client application server CLASSPATH would be something like:

```
-classpath C:\IBMVAGen\VGCSO45\hpt.jar;C:\MyProject\myClient.jar;
```

The EJB container sever application CLASSPATH would be something like:

```
-classpath C:\IBMVAGen\VGCSO45\hpt.jar;C:\MyProject\myServer.jar;
```

In these CLASSPATH examples, it is assumed that any property files required for the client to call a server program are included in myClient.jar and that any property files required for the session bean, and any local server programs it calls, are included in myServer.jar.

5. Click on the **Apply** push button.
6. Select the container in which the session beans are to be deployed and right mouse button click to bring up the container context menu.
7. Select **Create**, then **Enterprise Bean** to bring up a dialog in which you specify where to find the enterprise bean.
8. Click on the **Browse** push button to bring up a file selection dialog on the node on which your application server resides. Navigate to the exported session bean JAR file.

If you want to create enterprise beans individually, double click on your session bean JAR file, and a list of session bean deployment descriptors is displayed. Double click on the deployment descriptor file, and the session bean associated with the deployment descriptor will be deployed in the selected container.

If you exported an EJB Group, or just exported more than one session bean, you can deploy them all at once in the selected container. Just select the session bean JAR file, then click on the **Select** pushbutton. Then select **Yes** on the resulting confirmation message.

Invoking Generated Session Beans

Enterprise beans are typically called from servlets (or from beans used by the servlet) or from other enterprise beans. Using RMI in a Java application, you can invoke the remote interface methods of an enterprise bean deployed in a WebSphere AE or WebSphere EE enterprise server. However, this is not a preferred practice and will not be discussed here. The WebSphere enterprise Java server does not support invoking enterprise beans from applets.

There are two kinds of servlets that you can develop: stateless and stateful servlets. Stateless servlets do not save any data from one request to the next. Since stateless servlets are loaded once and reused by all clients making requests of that servlet, their code must be reentrant. Stateful servlets create a component called session data. A servlet that requests session data is not reused between clients. The servlet is reused by all requests from the same client, but is not shared between clients. When a stateful servlet receives a request, it generally asks for its session data. If it exists, saved information is retrieved. Otherwise, a new session data object is initialized. (Another option is that servlets save their state in a database and retrieve it based on some input key.)

In the context of invoking session beans generated by VisualAge Generator, there are essentially three places in the servlet code where you need to insert code referencing the session bean or VisualAge Generator support classes associated with the Common Services component of VisualAge Generator. The actual code necessary to perform the required function in each of these places depends on whether you use a generated program wrapper to call the session bean or whether you call the remote interfaces of the session bean directly. There are also some differences depending on whether you are developing a stateless or a stateful servlet.

init method
In the init method of the servlet you need to locate the session bean.

doGet or doPost method
In these methods you:

- Retrieve information from input parameters, session data, or both
- Set up parameters to be passed to the session bean
- Invoke the call method of the session bean
- Retrieve returned parameters
- Output the response HTML. A good practice is to invoke a JSP to output the response HTML

destroy method
In the destroy method of the servlet you need to tell the enterprise Java server that you are through with the session bean.

Invoking a Session Bean Using a Generated Program Wrapper

When you generate a session bean, wrappers are also generated for the server program and for its parameters. You can use a program wrapper to easily locate and call a generated session bean.

- In the init method of the servlet, create an instance of the CSOLocalPowerServerProxy class to set up for calling the Power Server API of the Generator. When you generate the program wrapper with a linkage table specifying that the server program is to be invoked from a generated session bean, the server program wrapper knows that it must tell the CSOLocal PowerServerProxy class to locate the session bean for the program, and that it must implement the calls using the remote interface of the session bean.

  If you are developing a stateful servlet, you may also want to create an instance of the server program wrapper, and ask the server program wrapper for each of its record parameter wrappers.
- In the doGet or doPost methods, you use the set() methods of the record parameters to set their values, then invoke the execute() method of the server program wrapper.

  If you are developing a stateless servlet and did not create an instance of the program wrapper in the init method, you would do that first. Remember that instance variables declared at the class level are shared between clients for stateless servlets. You would not want to share your server program data with another client.

  Parameter updates are placed in the record parameter wrappers. You can use the get() methods of the record wrappers to access their data.
- In the destroy method, call the close() method of the CSOLocalPowerServerProxy class you created in the init method. A good practice is to set the pointer to the CSOLocalPowerServer Proxy instance to null so that garbage collection can remove the instance from memory. If you also created an instance of the server program wrapper, you should set the reference to that instance to null.

Invoking a Session Bean Without Using a Generated Program Wrapper

If you want to use the elementary methodology for locating and calling session beans, you have to write more complicated code:

- In the init method of the servlet, you must locate the session bean using Java Naming and Directory Interface (JNDI) classes. Then you must get a reference to the remote interface of the session bean. See the ″Writing Enterprise Beans″ manual shipped with WebSphere AE for details on how to do this.
- In the doGet or doPost methods, you
  - Create an instance of each record parameter wrapper and use the set() methods of the record parameters to set their values,
  - Invoke the call() method of the remote interface of the session bean, assigning the return value to an array of Objects with an element for each parameter.
  - Retrieve updated parameters from the object array returned by the call() method of the session bean.

Objects are passed by value when calling enterprise bean remote interface methods. This means that updates to the parameters are made to copies of the input objects. Updates have to be made in a return value. Because generated session beans may make updates to each input parameter, and you can only return one object, the return value of the call() method of a generated session bean is an array containing an element for each input parameter. On return from the call() method you must assign the elements of the return array to their proper variables of your servlet.

  – Since session beans may be removed or deactivated by the enterprise Java server due to timeouts, the reference to the session bean you obtained in the init() method of your servlet may become invalid. See the ″Writing Enterprise Beans″ manual for details on how to code your servlet to adjust to this situation.

- In the destroy method,
  – Call the close() method of the session bean to close the communications session created by the session bean to call the server program. (Note that the server program may reside on a different machine than the one where the session bean is deployed.)
  – Call the remove() method of the session bean to tell the enterprise Java server that you are through with the session bean.
  – A good practice is to set your reference to the remote interface of the session bean to null.

Examples of Calling a Server Program In a Servlet Using a Session Bean

The following example shows the external source file specification of the interface for a simple server program that receives one record parameter. The record parameter contains one data item of type CHA. This program reverses the order of the characters of the data item.

```
:EZEE 440              10/29/99 12:10:27
:program   name      = ELACVP5
           date      = '05/28/1999' time = '17:11:03' type = CALLBATCH
           pfequate  = N     implicit = Y
           execmode  = NONSEGMENTED
:callparm  name      = ALLSTR
           type      = RECORD
:eprogram.
:record    name      = ALLSTR
           date      = '05/28/1999'        time = '17:11:12'
           org       = WORKSTOR
           usage     = SHARED
:recditem  name      = STRSTUFF
           level     = 10             occurs = 00001
           usage     = SHARED
:erecord.
:item      name      = STRSTUFF
           date      = '05/28/1999' time = '17:11:06' type = CHA
```

```
            bytes     = 00007      decimals = 00     evensql = N
:mapedits  fillchar  = 'N'
            inputreq  = N    justify = LEF
:eitem.
```

Example of Calling a Session Bean Using a Program Wrapper: The following example is a stateless servlet that calls the generated session bean for server program ELACVP5 using a server program wrapper to make the call.

```
package reversi.ejbs;
// EJB imports
import java.rmi.RemoteException;
// VA Generator Common Services imports
import com.ibm.vgj.cso.*;
// general Java imports
import java.util.*;
import java.text.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ReversiEJBWrapServlet extends HttpServlet
{
  private CSOLocalPowerServerProxy powerServer = null;

  // Constructor
  public ReversiEJBWrapServlet ()
  {
    super();
  }
  /** Close the wrapper for server program ELACVP5.
  * This will close the communications session to the server program.
  */
  public void destroy()
  {
    if (powerServer != null) // If created power server
    {
      try
      {
        powerServer.close();
        powerServer = null;
      }
      catch (Exception e)
      {
        System.out.println("Received exception in close: " + e.getMessage());
      }
    }
  }
/**
* Call session bean generated for server program ELACVP5 to reverse an input string
*/
public void doGet(HttpServletRequest req, HttpServletResponse res) throws IOException
{
  PrintWriter htmlOut = res.getWriter();  // HTML output stream
  Elacvp5 serverPgm = new Elacvp5(powerServer); // Wrapper for Server Program
  Allstr record = serverPgm.getAllstr();  // Input/Output Parameter
  String inputString = null;

  try
  {
    // Read input parameter from request object.
    inputString = req.getParameter("InputString");
    if (inputString == null)
```

```
        inputString = "1234567";  // Default input
      record.setStrstuff(inputString);
      // Call the server program using its wrapper
      serverPgm.execute();
      // Format output of server program. Should do this in a JSP
      htmlOut.println("Input string was <b>" + inputString + "</b>.
                       Output string is <b>" + record.getStrstuff() + "</b>.");
    }
    catch (CSOException e)
    {
      // Error in server program, or in communcation with server
      htmlOut.println("CSO exception: " + e.getMessage());
      System.out.println("Received CSO exception in call: " + e.getMessage());
    }
    catch (Exception e)
    {
      htmlOut.println("General exception " + e.getMessage());
      e.printStackTrace();
    }
  }
  /**
   * Create an instance of a power server that will use the VA Generator
   * Power Server API to communicate between the application server
   * and the server program.
   *
   * If the wrapper was generated with a linkage table entry indicating that
   * the server program is to be called from a session bean, the session bean
   * will be located, and the remote interface of the session bean will be used
   * to invoke the Power Server API from the enterprise bean server.

   * Otherwise, the Power Server API will be invoked from the Web application
   * server where this servlet is running.
   */
  public void init(ServletConfig servletConfig) throws ServletException
  {
    super.init(servletConfig);
    try
    {
      powerServer = new CSOLocalPowerServerProxy();
    }
    catch (CSOException e)
    {
      // Error setting up for communications with the Power Server API
      System.out.println("Received CSO exception: " + e.getMessage() );
    }
  }
  }
```

Example of Calling a Session Bean Without Using a Program Wrapper: The following example is a stateless servlet that calls the generated session bean for server program ELACVP5 without using a server program wrapper to make the call.

```
package reversi.ejbs;
// EJB imports
import java.rmi.RemoteException;
import java.rmi.NoSuchObjectException;
// VA Generator Common Services imports
import com.ibm.vgj.cso.*;
// general Java imports
import java.util.*;
import java.text.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ReversiEJBNoWrapServlet extends HttpServlet
```

```
{
  private Elacvp5EJB sessionBean; // Object implementing session bean's
                                  // remote interface
 /** Defaut constructor.
 */
 public ReversiEJBNoWrapServlet ()
 {
   super();
 }
 /**
  * Close the communications session establised by the session bean in order to call
  * the server program. Then remove the session bean from the its home.
  */
 public void destroy()
 {
   if (sessionBean != null) // If successfully got reference to the session bean
   {
     try
     {
       /* Close communications session between session bean server and */
       /* server program machine                                       */
       sessionBean.close();
       sessionBean.remove(); // Remove the session bean from its home
       sessionBean = null;
     }
     catch (Exception e)
     {
       System.out.println("Trying to close, received exception: " + e.getMessage());
     }
   }
 }
 /**
  * Call the session bean to reverse an input string.
  */
 public void doGet(HttpServletRequest req, HttpServletResponse res) throws IOException
 {
   PrintWriter out = res.getWriter(); // HTML output stream
   Allstr record = new Allstr(); // Input Parameter wrapper
   Object[] outputParms = null; // Output of session bean
   boolean goodRef = false; // Assume lost reference to session bean
   String inputString = null; // String to reverse
   // Read input parameter from request object.
   inputString = req.getParameter("InputString");
   if (inputString == null)
     inputString = "1234567"; // Default input
   // Set data item in input record wrapper
   record.setStrstuff(inputString);

   // Call the server program through the session bean. If the session bean
   // reference previously obtained has timed out, a NoSuchObjectException
   // will be thrown, a new session bean will be obtained, and this block
   // of code will be repeated.
   do
   {
     // Call the server program using a generated session bean
     try
     {
       outputParms = sessionBean.call(record);
       record = (Allstr) outputParms[0];
       // Format output of server program. Should do in JSP
       out.println("Input string was " + inputString +
                   ". Output string is " + record.getStrstuff() + ".");
       goodRef = true; // Found session bean, so fall through to exit
     }
     catch (NoSuchObjectException e)
     {
       // Reference to session bean is no longer valid
       try
       {
         sessionBean = locateSessionBean();
         goodRef = false; // so repeat call attempt
       }
       catch (CSOException ecso)
       {
         // Error in server program, or in communcation with server
         out.println("CSO exception: " + ecso.getMessage());
         System.out.println("Received CSO exception: " + ecso.getMessage());

         goodRef = true; // so fall through to exit
       }
```

```
      }
      catch (CSOException e)
      {
        // Error in server program, or in communcation with server
        out.println("CSO exception: " + e.getMessage());
        System.out.println("Received CSO exception: " + e.getMessage());
        goodRef = true; // so fall through to exit
      }
      catch (Exception e)
      {
        out.println("General exception " + e.getMessage());
        e.printStackTrace();
        goodRef = true; // so fall through to exit
      }
    } while (!goodRef);  // So try again if session bean reference no longer valid
  }
  /**
   * Obtain a reference to the session bean used to call the server program so
   * that its remote iterface can be used.
  */
  public void init(ServletConfig servletConfig) throws ServletException
  {
    super.init(servletConfig);
    try
    {
      sessionBean = locateSessionBean();
    }
    catch (CSOException e)
    {
      // Error setting up for communications with the Power Server API
      System.out.println("Received CSO exception: " + e.getMessage());
    }
  }
  /**
  * Locate the remote interface of the session bean generated for a server program.
  *
  * @return koch.test.reversi.ejbs.Elacvp5EJB - remote interface for session bean
  *                                            generated for server program ELACVP5
  */
  public Elacvp5EJB locateSessionBean() throws CSOException
  {
    String namingFactory = null;    // Naming factory to use to get initial context
    String beanJNDIName = null;     // JNDI name for session bean
    Elacvp5EJBHome beanHome;        // Narrowed home interface implementation class
    Elacvp5EJB beanRemoteInterface; // Object implementing session bean's remote interface

    try
    {
      // Get the initial naming context for resolving session bean location
      java.util.Properties properties = new java.util.Properties();
      /* If do not want to use a name server on the enterprise bean server, listening on port 900,
         set the PROVIDER_URL property to "iiop://hostname:port/"

         providerURL = "iiop://myhost:myport/";  // URL for the name server
         properties.put( javax.naming.Context.PROVIDER_URL, providerURL );
      */

      /* A different naming factory is required for ComponentBroker */
      namingFactory = "com.ibm.ejs.ns.jndi.CNInitialContextFactory"; // Naming factory for WebSphere AE
      properties.put( javax.naming.Context.INITIAL_CONTEXT_FACTORY, namingFactory );

      /* Get an initial contect to use to fine session bean home */
      javax.naming.InitialContext initContext = new javax.naming.InitialContext( properties );

      // Locate the server program's session bean home
      try
      {
        beanJNDIName = "Elacvp5EJB";   // JNDI name for the generated session bean
        // Get un-narrowed implementation of the home interface
        Object ejbHome = initContext.lookup(beanJNDIName);
        /* Get the narrowed object implementing the session bean's home interface */
        beanHome = (Elacvp5EJBHome)javax.rmi.PortableRemoteObject.narrow((org.omg.CORBA.Object)
                   ejbHome, Elacvp5EJBHome.class);
      }
      catch ( javax.naming.NamingException e )
      {
        // Error looking up home interface of session bean
        throw new CSOException(e);
      }
      catch (Exception e )
      {
        // Error looking up home interface of session bean
        throw new CSOException(e);
      }
    }
    catch ( Exception e )
    {
      // Could not get initial context from JNDI name server
      throw new CSOException(e);
    }
    try
    {
      beanRemoteInterface = beanHome.create();
    }
    catch ( Exception e )
    {
      // Could not get session bean's remote interface
      throw new CSOException(e);
    }
    return (beanRemoteInterface);
  }
}
```

Java Package Examples

How to Call Server Wrappers from Java Applications

A CSOLocalPowerServerProxy establishes a communication session with the VisualAge Generator PowerServer API for the purpose of calling VisualAge Generator server programs via the PowerServer API.

To call a server program named STAFFMN from a Java application, generate a JavaBeans wrapper class for the server program using VisualAge Generator (the Java class name will be Staffmn). For this example assume the generation option /PACKAGENAME=StaffPkg was specified. Then code the following in the Java application:

```
import com.ibm.vgj.cso.*;
import StaffPkg.*;   // Needed only if Java application not in package StaffPkg
    .
    .
   // Initialization
  CSOLocalPowerServerProxy powerServer = new CSOLocalPowerServerProxy();
  Staffmn staffmn = new Staffmn(powerServer);
  StaffMaint staffMaint = staffmn.getStaffMaint();
    .
    .
  // Processing. Many calls may be made without closing the PowerServer
  try
  {
      .
      .
    staffmn.execute();
      .
      .
    // Commit (use only if linkage table specifies client unit of work)
    powerServer.commit();
      .
      .
  }
  catch (CSOException error)
  {
    // Rollback (use only if linkage table specifies client unit of work)
    try
    {
      powerServer.rollback();
    }
    catch (CSOException rbError)
    {}
    String explanation = error.getMessage(); // Use to display error message
      .
      .
  }
  .
  .
  // Explicitly close the PowerServer unit of work when the application
  // ends.  Do not rely on garbage collection, which may leave the
  // session hanging for a long period of time.
  //
  if (powerServer != null)
  {
    try
    {
      powerServer.close();
      powerServer = null;     // So garbage collection can delete unit of work
    }
    catch (CSOException error)
    {
      String explanation = error.getMessage();
      .
      .
    }
  }
```

Multiple server programs can be called from the same PowerServer instance.

See com.ibm.vgj.cso.CSOLocalPowerServerProxy.html for more information.

How to Call Server Wrappers from Applets

A CSORemotePowerServerProxy establishes communication with a CSORemotePowerServerImpl object on a VisualAge Generator Java gateway using the Java 1.1 Remote Method Interface (RMI) for the purpose of calling VisualAge Generator server programs.

To call a server program named STAFFMN from an applet, generate a Java wrapper class for the applet using VisualAge Generator (the Java class name will be Staffmn). Then code the following in the applet:

```
          import com.ibm.vgj.cso.*;
import StaffPkg.*;   // Needed only if applet not in package StaffPkg
    .
    .
```

```
public void init()
{
  // Initialization
  try
  {
    // Establish communication with PowerServer on machine from
    // which the applet was loaded
    CSORemotePowerServerProxy powerServer =
       new CSORemotePowerServerProxy(this.getCodeBase().getHost());
  }
  catch (CSOException error)
  {
    String explanation = error.getMessage();
      .
      .
  }
  Staffmn staffmn = new Staffmn(powerServer);
  // Get the server program's record parameter
  StaffMaint staffMaint = staffmn.getStaffMaint();
    .
    .
 }
public void actionPerformed(ActionEvent event)
{
     .
     .
  // Handle action requiring server call
  try
  {
    // Processing
       .
       .  // Use set methods to set data items
       .
    staffmn.execute();
       .
       .
    // Commit (use only if linkage table specifies client unit of work)
    powerServer.commit();
       .
       .
  }
  catch (CSOException error)
  {
    // Rollback (use only if linkage table specifies client unit of work)
    try
    {
      powerServer.rollback();
    }
    catch (CSOException rbError) // Catch error on rollback
    {}
    String explanation = error.getMessage(); // Use to display error message
      .
      .
  }
  .
  .  // Use get methods to retrieve data item values
  .

  // Handle action to terminate applet
  //
  // Explicitly close the PowerServer unit of work when the applet is
  // destroyed. You may also want to close the unit of work when the
  // applet is hidden and create a new one if made visible again. If
  // using a client unit of work, you would need to be sure there is
  // no outstanding work before closing. Do not rely on
  // garbage collection, which may leave the session hanging for a
  // long period of time.
  //
  if (powerServer != null)
  {
    try
    {
      powerServer.close();
      powerServer = null;     // So garbage collection can delete unit of work
    }
    catch (CSOException error)
    {
      String explanation = error.getMessage();
       .
       .
    }
  }
}
```

Multiple server programs can be called from the same remote power server.

See com.ibm.vgj.cso.CSORemotePowerServerProxy.html for more information.

How to Run Applets From a Browser

To run an applet, you need to reference the applet in an HTML file you download from your Web browser. Here is the text of a minimal HTML file (named Staffmn.html) for running an applet:

```
<APPLET code="StaffmnApplet.class" width=400 height=400>
</APPLET>
```

Note: Your browser must be capable of running Java 1.1 applets, including Remote Method Invocation (RMI), to work with VisualAge Generator wrapper classes.

To run your applet from a Java SDK applet viewer, enter the following command:

```
appletviewer http://hostname/hostalias/Staffmn.html
```

Where **hostname** is the network identifier of your VisualAge Generator Java gateway system and **hostalias** is the alias by which the directory on which Staffmn.html resides is known to the VisualAge Generator Java gateway.

For OS/2 the command to start the applet viewer in your Java SDK may be `applet` instead of `appletviewer`.

How to Start the Session Manager on your VisualAge Generator Java gateway

If you are using a browser or a Java applet viewer to download an applet from a VisualAge Generator Java gateway, the Session Manager must first be started on the VisualAge Generator Java gateway machine. To start the Session Manager, your VisualAge Generator Java gateway system must be at Java Version 1.1 or higher.

When the Session Manager is started, it registers a RemotePowerServer with the Java Remote Method Invocation registry for the purpose of calling the PowerServer API on behalf of Java applets. The Remote Method Invocation registry must be started before starting the Session Manager.

To start the JAVA Remote Method Invocation registry on OS/2 or Windows NT host systems, enter the following:

```
start rmiregistry
```

To start the JAVA Remote Method Invocation registry on AIX or OS/390 host systems, use the following background command:

```
rmiregistry &
```

On OS/400, you need to start the QVGNSBS subsystem if it has not been started on the AS/400 machine. To do so, enter the following command:

```
STRSBS SBSD(QVGNSBS)
```

To start the Java Remote Method Invocation registry on AS/400 host systems, use the following background command:

```
SBMJOB CMD(QSH CMD('rmiregistry')) +
JOB(VGNRMI) +
JOBD(QPGMR) +
JOBQ(*LIBL/QVGNSBSQ) +
CPYENVVAR(*YES)
```

Make sure that your CLASSPATH environment variable is set properly and includes both hpt.jar and jt400.jar.

To start the Session Manager on OS/2 or Windows NT for handling calls from applets to VisualAge Generator server programs, enter the following command:

```
java com.ibm.vgj.cso.CSOSessionManager options
```

See “Specifying Session Manager Options” on page 339 for details on how to set session manager options from the command line.

To start the Session Manager on AIX or OS/390 for handling calls from applets to VisualAge Generator server programs, enter the following command:

```
java com.ibm.vgj.cso.CSOSessionManager options &
```

To start the Session Manager on AS/400 for handling calls from applets to VisualAge Generator server programs, enter the following command:

```
SBMJOB CMD(RUNJVA CLASS(CSOSessionManager) PARM('--nogui') CHKPATH(*IGNORE)) +
       JOB(VGNSMGR) +
       JOBD(QPGMR) +
       JOBQ(*LIBL/QVGNSBSQ) +
       CPYENVVAR(*YES)
```

As an alternative, you can enter the following command:

```
CALL PGM(QVGEN/QVGNSMGR)
```

This is a CL program that sets up the rmiregistry if it is not started already, and runs CSOSessionManager with −−nogui specified.

Specifying Session Manager Options

Session manager options may be obtained:

1. From default values.
2. From a Java properties file named sessionmanager.properties that is searched for in the CLASSPATH in effect when the session manager is started. This properties file is placed into the directory where the Common Services component is installed (default is C:\IBMVAGEN\VGCSO45). Its option values are the default values, but can be changed by a system administrator.
3. From command line arguments specified when starting the session manager.
4. From the session manager administrative user interface.

Options are overriden in the sequence they occur in this list. For example, a command line argument specification for an option overrides a properties file specification for that same option.

Session manager options may be specified for the following:

- Whether or not an administrative GUI is to be displayed on the VisualAge Generator Java gateway (except on OS/400)
- How long client sessions can remain inactive before their resources are freed, including their connection to server program machines.
- What gets traced and where the trace output is sent.
- What gets logged and the name of a log file.

The following table describes the options that may be specified, the syntax for command line specification of the option, the property name to be used for the option in a properties file, the valid values, and the default value. Property files use (property name, property value) pairs to define options. For the sessionmanager.properties file, all property names in the table below must be prefixed by **cso.sessionmanager.**. For example, to indicate that inactive sessions are to be closed after 30 minutes, specify a property name of cso.sessionmanager.SessionCheckInterval and a property value of 30.

Table 43. Session Manager options

Option Function	Argument Name	Property Name (Prefixed with cso.sessionmanager.)	Value Domain	Default Value
Indicator of whether to display GUI	--gui, --nogui	Gui	TRUE, FALSE in properties file (No value for command line arguments. Implied by --gui and --nogui)	TRUE (--gui)
Maximum time session may be inactive, in minutes	--checkInterval	SessionCheckInterval	0<Integer<1441	1440 (one day)
Trace level	--traceLevel	trace.Level	**0** Trace nothing **1** Trace errors **2** Trace requests to Gateway **4** Trace parameters before and after call **8** Trace call options **Sum of any above** For example, 15 (8+4+2+1) means trace all	0

Table 43. Session Manager options (continued)

Option Function	**Argument Name**	**Property Name (Prefixed with cso.sessionmanager.)**	**Value Domain**	**Default Value**
Trace output type	--traceType	trace.Type	**STDOUT** Write output to STDOUT **STDERR** Write output to STDERR **FILE** Write output to a file specified by trace specification **WINDOW** Write output to a window that is opened to trace session or program activity	
Trace output details: • Name of file if trace type is FILE • Not applicable otherwise	--traceSpec	trace.Spec	valid file name or URL	null string
Log level	--logLevel	log.Level	**0** Log nothing **1** Log errors **2** Log requests to Gateway **4** Log parameters before and after call **8** Log call options **Sum of any above** Trace all used in sum. For example, 15 (8+4+2+1) means trace all	0

Table 43. Session Manager options (continued)

Option Function	Argument Name	Property Name (Prefixed with cso.sessionmanager.)	Value Domain		Default Value
Log output type	Not applicable. Always FILE	log.Type	**FILE**	Write output to a file specified by log specification	FILE
Name of log file	--logSpec	log.Spec	valid file name		CSOJava.log

Java Support for OS/390 Unix Systems

VisualAge Generator JavaBeans wrapper support allows you to use a web browser to access a Java applet residing on an OS/390 Unix Systems VisualAge Generator Java gateway. The applet uses generated JavaBeans wrappers that call the Power Server API to invoke a CICS transaction. This is a two-tiered solution to access an MVS CICS server program rather than requiring a gateway between the web browser and the MVS CICS system. Because the External CICS Interface (EXCI) is the only communications protocol provided by the Power Server API on OS/390 Unix Systems, the OS/390 VisualAge Generator Java gateway cannot be used as a middle tier in a three-tier system other than accessing various MVS CICS regions.

Linkage Table Entries for OS/390 Java Support

There are no differences to the way a Java applet for OS/390 interfaces with the generated wrappers. There are also no differences in the process of generating the JavaBeans wrappers for a server program. There is, however, an enhancement to the linkage table REMOTECOMTYPE parameter to provide support for OS/390.

To successfully invoke a server program using the VA Generator Java support for OS/390, the linkage table used to specify linkage parameters must specify:

- LINKTYPE=REMOTE
- REMOTECOMTYPE=EXCI

  EXCI is a new value for REMOTECOMTYPE that is valid only for Java support for OS/390. When specified, it causes the Power Server APIs to access MVS CICS using the External CICS Interface.
- CONTABLE=CSOExxxx

  where xxxx is the conversion table suffix for the language installed where the server program is to execute. See Table 55 on page 436 for valid conversion table names.
- LOCATION=cicsRegion

where cicsRegion is the CICS region where the server program is to run.

- SERVERID=transactionID

  where transactionID is the ID of the transaction used to invoke the server program. The transaction definition must specify the mirror program DFHMIRS as the initial program, and the profile DFHCICSA. (This is an EXCI restriction.)
- LUWCONTROL=SERVER

Extended (client) unit of work is not yet supported by the EXCI.

MVS CICS Setup for EXCI

Details on usage of the External CICS Interface can be found in the CICS for MVS/ESA External CICS Interface (SC33-1390) and the CICS Internet and External Interfaces Guide (SC33-1944) documents. These documents provide information on defining connections and sessions, security considerations and setup, and return codes.

Defining Connections and Sessions for EXCI

In order to use the Java support on OS/390 Unix Systems, you must define CICS connections and sessions for the CICS group installed to run the transactions requested from Java. In MVS CICS 4.1, the External CICS Interface allows only 25 simultaneous requests per connection. In order to handle more than 25 simultaneous requests, the EXCI communications driver will use up to 100 connections with predefined names. You must define a connection and session for each multiple of 25 simultaneous requests that are to be handled. For each connection you must define a CICS connection as follows:

For each connection, you must define a CICS connection as follows:

- Netname = ELA000xx

  where xx is a two digit number between 00 and 99. This name is also used in EXCI security checks.
- Access method = IRC
- Protocol = EXCI
- Connection type = SPECIFIC

For each connection you must define a CICS session as follows:

- Connection = connectionName

  where connectionName is the name of the connection for which the session is being defined.
- Protocol = EXCI
- Send count = blank

Once the group that contains these connections is installed, you must ensure that the interregion communication (IRC) is open. If IRC is not opened during CICS initialization, set it to open using the CEMT SET IRC OPEN command.

External CICS Interface Security

The External CICS Interface provides very little security checking. The main security checking must be performed in the Web server that services html requests. Lotus Domino Go Server is one product that enables you to provide security checks for accessing html pages.

There is no password support in the EXCI. However there are some security checks made based on the connection name used to invoke the EXCI, and, optionally, on a user ID, depending on whether the CICS system being used is configured to do security checking.

The External CICS Interface does provide security checking for the following:

- Logon Security
  - The Netname attribute of a connection used to invoke the EXCI (ELA000xx for the OS/390 Java support) must be defined as a user profile to RACF.
  - The Netname attribute of a connection used to invoke the EXCI must be authorized to its own DFHAPL.net_name RACF FACILITY class profile, with UPDATE authority.
  - The Netname attribute of a connection used to invoke the EXCI must be authorized to the DFHAPPL.location RACF FACILITY class profile of the target CICS server region (the value of the location attribute for the server program's linkage table entry) with READ authority.
- Link Security

  The connections used by the EXCI must be authorized to the following CICS resource profiles:
  - The profile for the mirror transaction specified in the serverId attribute of the server program's linkage table entry.
  - The profile for all the resources accessed by the server program.
  - The CICS command profiles for any SPI commands issued by the CICS server program.
- User Security

  If a user ID is passed using the setUserId method of the CSOClientCallOptions class, and the CICS server region specified user security checking is to be performed (using ATTACHSEC(IDENTIFY)), the user ID passed must be authorized to the resources described for link security, in addition to the authorization for link security.

Graphical User Interfaces and OS/390

In order to implement a GUI on OS/390 Unix Systems, you have to use a distributed display to an XWindows server through TCP/IP. Implementing a GUI requires that you:

1. Authorize the OS/390 Unix Systems machine to use the XWindows server. This can be done on the XWindows server with the following command:
   ```
   xhost +OS390_Server_Address
   ```
2. Set the DISPLAY environment variable on the OS390 Unix Systems session to direct the display to an XWindows server port. For example,
   ```
   export DISPLAY=9.37.200.100:0
   ```

   where 9.37.200.100 is the IP address of the XWindows server, and 0 is the port of the monitor on which the GUI is to be displayed.

In order to use the VisualAge Generator Java support to run a Java GUI application calling a server program on OS/390 Unix Systems, you must direct the GUI to an XWINDOWS server.

The GUI of a Java applet runs on the Java client, not on the OS/390 gateway. However, the Session Manager by default displays an administrative user interface. On OS/390, to use this function you must direct the OS/390 VisualAge Generator Java gateway display to an XWindows server as described above. Because you may not want to require an XWindows server, a command line argument, --nogui, allows you to start the RemotePowerServer on OS/390 without an administrative user interface. For example, to avoid using the Session Manager administrative user interface, you would start the Session Manager with the following command:

```
java com.ibm.vgj.cso.CSOSessionManager --nogui &
```

If you do not direct output to an XWindows server, then the capabilities of the session manager administrative user interface are lost. Sesssion manager options are taken from the session manager properties file and/or command line arguments, and cannot be changed without stopping the Session Manager and restarting it with new options.

To stop the Session Manager in OS/390 Unix Systems, you can use the Unix ″kill″ command to end the Session Manager's process. Occasionally you may need to start a new OpenEdition shell without exiting the session in which you started the RemotePowerServer. To do this enter subcommand mode by pressing the escape key and entering ″open″ to start a new shell process. To return to the previous session, enter the ″exit″ or ″PrevSess″ command. To get out of subcommand mode in the original session, enter the ″return″ command.

Starting the Session Manager Using JCL

Instead of starting the RMI registry and the Session Manager from the shell command line, you can start them through JCL that invokes the BPXBATCH program shipped with OS/390 Unix Systems. The BPXBATCH program is documented in manual SC28-1892, OS/390 OpenEdition Command Reference. The time parameter on the job card and the exec statement may allow you to avoid time constraints imposed by your system when starting processes from the command line.

To use the BPXBATCH program to start the RMI registry and the Session Manager, you have to:

- Create a file used to set environment variables required by the RemotePowerServer.
- Develop JCL to execute the BPXBATCH program with the proper file allocations.

Setting Environment Variables for BPXBATCH

BPXBATCH uses an input file to set environment variables that would be set from your .profile dataset when starting processes from the Open Edition shell command line. This file must be allocated to ddname STDENV. An example of such a file follows:

```
HOME=/u/myuserid
PATH=$PATH:$HOME:/usr/lpp/java/J1.1/bin
LIBPATH=/u/myuserid/ServerPrograms:/usr/lpp/vgwgs:$LIBPATH
STEPLIB=CICS.CICS410.SDFHEXCI:$STEPLIB
DPATH=/u/myuserid/ServerPrograms:/usr/lpp/vgwgs:$DPATH
CLASSPATH=/u/myuserid/public:/usr/lpp/vgwgs/hpt.jar
CSOLINKTBL=/u/myuserid/ServerPrograms/LINKSMVS.LKG
export PATH LIBPATH DPATH CLASSPATH STEPLIB CSOLINKTBL HOME
```

Shell Script for Starting Servers from BPXBATCH

You can have BPXBATCH execute a shell script by allocating the shell script file to ddname STDIN. BPXBATCH can execute a shell script to start the RMI server and the Session Manager. Both servers can be started as background jobs that remain running after the job itself ends. An example of a shell script follows:

```
echo HOME=$HOME
echo PATH=$PATH
echo DPATH=$DPATH
echo CLASSPATH=$CLASSPATH
echo LIBPATH=$LIBPATH
echo STEPLIB=$STEPLIB
echo CSOLINKTBL=$CSOLINKTBL
cd /u/myuserid/public
echo Current directory is $(pwd)
nohup rmiregistry &
```

```
sleep 2
nohup java com.ibm.vgj.cso.CSOSessionManager --nogui &
echo Processes after start Session Manager
ps -ef
```

In this example the environment variable settings are ″echoed″ to record their values. (This is not required.) The output is sent to the file allocated by ddname STDOUT. The ″sleep 2″ command waits 2 seconds to allow the RMI registry to completely start before the Session Manager gets started. The ″ps -ef″ command records the process id's of the server processes in case you need to cancel them using the ″kill″ command.

The statement starting the Session Manager uses the --nogui option because there is no graphical display available in the batch environment. Options are taken from command line arguments and/or the session manager properties file. If you do not want the job used to start the servers to end, remove the ″nohup″ from the beginning and the ″&″ from the end of the statement starting the Session Manager.

JCL for Starting Servers from BPXBATCH

The JCL to start the RMI server and the Session Manager allocates the shell script to ddname STDIN, the file setting environment variables to STDENV, the file for standard output to ddname STDOUT, and the file for standard error to ddname STDERR. Following is an example of ″pseudo-JCL″ used to invoke BPXBATCH to start the RMI server and the Session Manager.

```
//jobname JOB (,,,,),user,TIME=1440,NOTIFY=user,MSGCLASS=T
//*============================================================*

//* JCL To start the RMI and VAGen Java Gateway Session Manager *
//*============================================================*

// EXEC PGM=BPXBATCH,TIME=1440
//STEPLIB DD DSN=CICS.CICS410.SDFHEXCI,DISP=SHR
//STDENV DD PATH='/u/user/BPXBATCH.env',PATHOPTS=ORDONLY
//STDIN DD PATH='/u/user/public/stpwrsrv.scr',PATHOPTS=ORDONLY
//STDOUT DD PATH='/u/user/javastd.out',
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),PATHMODE=SIRWXU
//STDERR DD PATH='/u/user/javastd.err',
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),PATHMODE=SIRWXU
```

Java Support for AS/400 Servers

VisualAge Generator supprts JavaBeans wrappers. This lets you use a Web broswer to access a Java applet residing on an AS/400 VisualAge Generator Java gateway. The applet uses a generated JavaBeans wrapper that calls the Power Server API to invoke an AS/400 Server program. The server program can reside on the same machine as the gateway or on a remote AS/400 machine. This provides a two-tiered solution to access a remote AS/400 server program rather than requiring a separate gateway between the Web browser and the AS/400 system where the server program resides.

Because the Java400 protocol is the only communications protocol provided by the Power Server API on AS/400, the AS/400 VisualAge Generator Java gateway cannot be used as a middle tier in a three-tiered configuration to access a remote server other than an AS/400 server.

User Authentication

If the server program accesses files or a relational database, the user identified on the client must be authorized to run the server program and to access any files or relational tables using dynamic SQL statements.

You must code the Java GUI application to supply a user ID and password for user authentication to the remote AS/400 machine. You can either supply these values statically within the Java GUI application or code a prompt to ask the user for this information at run time. On the first server call a connection that uses these values will be made. This connection to the remote AS/400 machine will remain until the end of application execution. As long as the Java GUI application runs, any additional calls to that same remote AS/400 machine will use the user ID and password values supplied when the client first connected to the AS/400 server.

Linkage Table Entries for AS/400 Java Support

Java applets for AS/400 interface with generated wrappers no differently than Java applets for other hosts do. The process for generating JavaBeans wrappers for a server program is also no different for the AS/400. However, only the Java400 protocol is available to connect to an AS/400 server a Java applet that resides on an AS/400 Web server.

To invoke a server program using VisualAge Generator Java support for AS/400, the linkage table that specifies linkage parameters must contain the following values:

LINKTYPE=CSOCALL

REMOTECOMTYPE=Java400
: Java400 is a new protocol for REMOTECOMTYPE that is valid only for a Java client application to access an AS/400 machine. This uses the AS/400 Toolbox for Java to communicate to the remote AS/400 machine.

CONTABLE=CSO*xxxx*
: where *xxxx* is the conversion table suffix for the language installed where the server program is to run. For valid conversion table names, see Table 55 on page 436.

LOCATION=*as400_name*
as400_name is the name of the AS/400 machine where the server program is located and to be run.

LIBRARY=*library*
library is the name of the library where the server program is located.

LUWCONTROL=*type*
type is SERVER or CLIENT.

You can also supply this information in the linkage information properties of VAGenProgramPart on the free-form surface.

If your applet was generated with the **callink** attribute of remotebind set to RUNTIME or a protocol property under linkageInfo set to Runtime Bind, you need to update the vgj.properties file that the applet uses. Specify the linkage information for the program to be called. Java400 is the only allowed remotecomtype value for the Java applet. For example:

```
cso.serverLinkage.MYPROGRM.remotecomtype=Java400
```

For more information about the vgj.properties file, see the Java server information in the *VisualAge Generator Generation Guide*.

Graphical User Interfaces and AS/400

Java GUI applications cannot run directly on the AS/400 JVM, because no local graphic display capability exists on AS/400. This means that CSOSessionManager can be invoked only with the −−nogui option when running on an AS/400 computer. Thus, the features of the Session Manager administrative GUI are not available on the AS/400.

Generated Java applets can reside on the AS/400 and be retrieved and executed by the client browser.

Requirements for Client and AS/400 Server

In order to use the Java400 protocol in your Java GUI application, you must include the AS/400 ToolBox for Java (jt400.jar) in your CLASSPATH environment variable in addition to the VisualAge Generator Java run-time jar files. AS/400 Toolbox for Java is available from the following sources:

- Use **ftp** from a remote AS/400 machine with V4R2 or higher. It should reside at /QIBM/ProdData/HTTP/Public/jt400/lib/jt400.jar. When you are using **ftp** to get this file, make sure you are retrieving it as a binary file.
- Download it from: http://www.as400.ibm.com/toolbox/downloads.htm.

In order to use this feature, your AS/400 server machine (the machine at where your server program resides) must have VisualAge Generator Server for AS/400 installed. In addition to this, your ITF machine must meet the requirements for AS/400 Toolbox for Java: AS/400 Host Services at the right level of PTF and TCP/IP Connectivity Utilities for AS/400.

For more information, see the AS/400 Toolbox for Java programming information at the AS/400 Web site under “Setting up AS/400 Toolbox for Java.” The URLs for viewing or downloading the information are as follows:

```
http://www.as400.ibm.com/toolbox/index.htm
http://www.as400.ibm.com/toolbox/downloads.htm
```

No additional setup is necessary on the AS/400 server machine.

Deploying Java Classes

Once you have developed and compiled your Java applet and your generated JavaBeans wrappers, and have developed an html file that browsers will use to download your applet, you must deploy them in your run-time environment. Because all browsers do not have the same capabilities for downloading classes, the setup required on the VisualAge Generator Java gateway depends on which browsers have to be supported.

For security reasons Java restricts unsigned applets to accessing classes only from subdirectories of a ″codebase″ directory specified in your html file. The default codebase is the directory containing the html file. You can specify a different codebase using the CODEBASE attribute of the <applet> tag in your html file.

Deploying Classes Without Using Archive Files

Java class definitions generally contain a package statement that defines the Java package to which the class belongs. (Class names must be unique within a package.) Generated wrappers for programs are assigned the package specified by the /PACKAGENAME generation option. A package name consists of groups of characters separated by periods. When the classes of the package are deployed, and Java archive files are not used, the classes of the package must be contained in the last subdirectory of a chain of subdirectories whose names match the groups of characters in the package name. This chain of subdirectories must be relative to the codebase directory specified in the html file. Because the developed classes access classes in the com.ibm.vgj.cso and the com.ibm.vgj.util packages, the classes of these packages must reside in subdirectories /com/ibm/vgj/cso and com/ibm/vgj/util respectively, relative to the codebase directory. The codebase directory must also be in the CLASSPATH environment variable when starting the RMI registry and the Session Manager.

For example, suppose you are deploying the classes for the Staffmn example, and you are letting the codebase default in the html file accessed. If

- Your VisualAge Generator Java gateway is on an OS/2 or Windows machine
- the codebase directory is c:\codebase and
- the html file is Staffmn.html

then you should have a directory/file structure like the following:

- c:\codebase
  - Staffmn.html
  - StaffPkg (assuming /PACKAGENAME=StaffPkg was specified)
    - wrapper class files generated and compiled for the StaffPkg package
    - any classes implementing the GUI invoking the Staffmn server program if you assigned them to the StaffPkg package. If you did not assign these classes to the StaffPkg package, then you must have subdirectories relative to c:\codebase for the package you did assign them to.
  - com
    - vgj
      - cso subdirectory
        - class files for the com.ibm.vgj.cso package
      - util subdirectory
        - class files for the com.ibm.vgj.util package

You can extract the classes of the com.ibm.vgj packages into the directories above by:

1. Changing to the c:\codebase directory
2. Running the Java command:

   ```
   jar xvf c:\IBMVAGEN\VGCSO\hpt.jar
   ```

   assuming VisualAge Generator Common Services are installed in c:\IBMVAGEN\VGCSO. There are other packages included in hpt.jar than those needed for JavaBeans wrappers. You can delete subdirectories added except for the com.ibm.vgj.cso and the com.ibm.vgj.util packages. If you are also deploying 4GL Java GUIs using the same method of deployment, you cannot remove any packages. The other packages would be required for the 4GL GUIs.

This example shows the simplest way to deploy your developed classes and the com.ibm.vgj packages so that your applet can be downloaded without being signed. All browsers should support this technique of deployment.

Deploying Classes Using Archive Files

If the browsers expected to be used to download your applet support downloading Java archive (.jar) files, you can compress all classes (and any other accessed resources, like image files) into a jar file. This allows all required files to be compressed and downloaded with one request, reducing network traffic. To tell a browser that your applet and its associated classes reside in a jar file, use the ARCHIVE attribute of the <applet> tag of your html file.

For example, suppose you copy hpt.jar to a base directory used as the codebase for all your applets, say c:\codebase. Then you could use the export function of VisualAge for Java to create a jar file for the package containing your applet and the generated wrappers in directory c:\codebase.

To make your browser download your classes and the com.ibm.vgj.cso and com.ibm.vgj.util packages specify

```
CODEBASE=codebase URL
Where codebase URL is a URL that maps to the c:\codebase directory.
```

and specify

```
ARCHIVE="hpt.jar,staffmn.jar"
```

as attributes of the tag in your html file.

All jar files to be downloaded must reside in the codebase specified in your applet's html file (or defaulted to the directory containing the html file), or in a subdirectory of the codebase directory.

Some browsers throw a security manager exception if more than one jar file is specified on the ARCHIVE attribute. If your browser does this, you have to compress both the com.ibm.vgj.cso and the com.ibm.vgj.util package classes and your developed classes into one jar file. You can extract the classes of the com.ibm.vgj.cso and the com.ibm.vgj.util package into subdirectories relative to some temporary directory as described in section "Deploying Classes Without Using Archive Files" on page 350, then compress all classes needed into one jar file by switching to the temporary directory and using a command like the following:

```
jar cvf staffmn.jar StaffPkg\*.class com\ibm\vgj\cso\*.class com\ibm\vgj\util\*.class
```

If you are also deploying 4GL Java GUIs using the same method of deployment, you cannot remove any packages. The other packages would be required for the 4GL GUIs.

Applet Session Manager

When you start a Session Manager as described in How to Start a Session Manager on Your VisualAge Generator Java gateway, a Session Manager administrative GUI is started that lets you manage sessions started by Java clients sending requests to the RemotePowerServer. The Session Manager administrative GUI is not available for AS/400.

The Session Manager administrative GUI displays a list of all active sessions and a list of server programs that have been called. Users may add additional server program names to the list if they want special tracing to occur for a server program even if the server program has not been called yet.

From the administrative GUI you may also perform the following tasks:
- Change session manager options
- Display details about a selected session
- Display statistics about a selected server program
- Cancel sessions with rollback of work in progress
- Enable tracing a session’s activity
- Enable tracing calls to a server program

Monitoring Active Sessions

The session manager displays a list of active sessions in its Sessions pane. Each entry in the list displays:
- The session ID for the session. This can be used to correlate with information in session traces.
- The date/time the session was last active.
- The current status of the session.

You can display statistics for an active session by selecting the active session in the session list, clicking on the Sessions drop-down menu, then clicking on the Details menu item. Statistics shown are :
- Session ID.
- Time session was last active.
- Current status of the session. Possible status values are:

  between calls
  No Power Server request is active.

  active call
  A call to a server program is in progress.

  active commit
  A call to commit is in progress.

  active rollback
  A call to rollback is in progress.

closing
A call to close is in progress.

closed The session has been closed.

- Add Number of commits and rollbacks to the Session View details.
- Number server program, commit, and rollback calls.

Monitoring Called Programs

The session manager displays a list of programs that have been called since the session manager was started in its Called Applications pane.

You can display statistics for a server program by selecting the server program in the Called Applications list, clicking on the **Called Applications** menu, then clicking on the **Details** menu item. Statistics shown are :

- The date/time when the server program was first called after the session manager was started.
- The date/time when the server program was last called.
- The session ID of the last session from which the server program was called.
- The number of calls made to the server program.
- The average response time for all calls, from the perspective of the Java gateway.
- A list of sessions that currently have active calls to the server program.

Canceling Sessions

Sessions can be canceled, with a rollback of work in progress, as a result of the following:

- Being inactive for a specified amount of time

  When you start the Session Manager you may specify the amount of time a session may be inactive before it is canceled. This time may be changed using the **Settings** window.
- Explicit action against a selected session in the Sessions list.

  You may cancel an active session in the session list by selecting the session, clicking on the **Session** menu, then clicking on the **Delete** menu item.
- You may cancel all active sessions in the session list by clicking on the Session drop-down menu, then clicking on the Flush Sessions menu item.
- Clicking on the **Delete** button on the details view for a session.

Tracing Sessions

You may start a trace for a session by opening a session details view for the session, clicking on the **Session** drop-down menu, then clicking on the **Trace** menu item. This causes a **Trace Options** dialog to be opened where you can specify what trace entries you want, and where you want trace output to be placed.

Where trace entries are to be placed is initially set as shown in the **Settings** window. You can change where the trace output is to be placed by clicking on the **Trace Type** drop down list symbol, then selecting the desired output type.

If you choose to have the trace output written to a file, an entry field is added where you must provide the file name.

If you choose to have the trace data written to a window, a trace window is opened where trace entries are displayed. At any time you can clear the trace window by clicking on the Clear button. You can save the trace data to a file by clicking on the **Save as** button, then specifying the file name on the resulting file dialog.

A session trace window remains open even after a session is closed in case you need to get back to the trace details. To close the trace window use the Close menu item of the system icon.

The **Trace Options** window is initialized with the types of trace data to be written set just as in the Settings window. You can change which types of trace data are to be written by checking the desired **Trace Options** check boxes.

See "Specifying Session Manager Options" on page 339 for details about trace options.

Once you finish specifying the desired trace options, click on the **Accept** button to cause them to take effect.

Tracing Server Programs

You may start a trace for a server program by opening a Called Application details view for the program, clicking on the **Called Application** drop-down menu, then clicking on the Trace menu item. This causes a **Trace Options** dialog to be opened where you can specify what trace entries you want, and where you want trace output to be placed.

Where trace entries are to be placed is initially set as shown in the **Settings** window. You can change where the trace output is to be placed by clicking on the **Trace Type** drop down list symbol, then selecting the desired output type.

If you choose to have the trace output written to a file, an entry field is added where you must provide the file name.

If you choose to have the trace data written to a window, a trace window is opened where trace entries are displayed. At any time you can clear the trace window by clicking on the **Clear** button. You can save the trace data to a file by clicking on the Save as button, then specifying the file name on the resulting file dialog.

A program trace window remains open even when no active sessions are open that use the program in case you need to get back to the trace details. To close the program trace window use the **Close** menu item of the system icon.

The **Trace Options** window is initialized with the types of trace data to be written set just as in the **Settings** window. You can change which types of trace data are to be written by checking the desired **Trace Options** check boxes.

See "Specifying Session Manager Options" on page 339 for details about trace options.

Once you finish specifying the desired trace options, click on the **Accept** button to cause them to take effect.

Adding a Program to the Called Application List

The first time a server program is called, an entry for that program is added to the Called Application list. However, if you need to get trace information for the first call to a server program, you have to explicitly add the server program to the Called Application list before the first call. To do this:

1. Select the New menu item from the Called Application drop-down menu.
2. On the resulting dialog enter the name of the server program you want added to the list.
3. Click on the OK button.

Once the server program is in the Called Application list, you can select it and set tracing options for it as described in "Tracing Server Programs" on page 355.

Setting Session Manager Parameters

See "Specifying Session Manager Options" on page 339 for a description of all session manager options, and for details on how initial values are determined. The following session manager options may be changed using the **Settings** window:

- Inactive session timeout interval
- Where trace output is to be written
- The kind of data to trace
- The kind of data to log
- Log file name

To open the **Settings** window, from the Session Manager main window, click on the **File** drop-down menu, then click on the **Settings** menu item.

Timeout Interval

The inactive session timeout interval controls how often the session manager checks for inactive sessions. If a session has not had any activity since the previous check for inactive sessions, the session is canceled with a rollback of any work in progress.

Trace Options

Use the Trace Options check boxes to select the kind of trace data you want written. The trace data selected from the Settings window will be written for all interaction with the Power Server, not just for a particular session or server program. To restrict trace data to a particular session or a particular server program, see Trace Sessions or Trace Server Programs, respectively.

The possible choices are:

Trace Errors
Adds a trace entry when an error occurs for a Power Server request.

Trace Requests
Adds a trace entry when each Power Server request is received, and an entry when the request completes.

Trace Parameters
Adds a trace entry containing the data item values for each parameter passed on each call to a server program. Values for each parameter are written as a hex string in the code page of the target system. If the parameter is a record, there is one hex string containing the values of all the record's data items.

Trace Call Options
Adds trace entries containing the values of each call option for each call to a server program.

Log Options

Use the **Log Options** check boxes to specify the types of trace entries that are to be written to a log file. The possible types of trace entries are the same as for Trace Options.

Information is written to the file specified in the **Log Filename** entry field. Default log file name is CSOJava.log in the directory where the Session Manager is started.

Java Names

The names of classes, methods, and objects in the generated class definitions are derived from VisualAge Generator member names according to the following algorithm:

- Convert VisualAge Generator name to lowercase.

- Delete any dashes (-) or underscores (_) and change the character that follows the dash or underscore to uppercase.
- When the converted name is used as a class name or within a method name, translate the first character back to uppercase.
- Package name for generated objects is the value specified in the /PACKAGENAME generation option. The specified package must exist when generation is requested.

The following names are not supported by JavaBeans wrapper class generation:

- DBCS names
- Names that are Java reserved words. For example ″CLASS″.

The following table shows an example of the derivation of JavaBeans wrapper names from VisualAge Generator member names.

Table 44. Derivation of Java names from VisualAge Generator names

VisualAge Generator Part Type	Server (called batch) program name	Record name for record parameter	Level-77 item parameter	Name of sub-structured, multiply occurring item in record	Low level item in record or record array
VAGen Part Name	STAFFMN	STAFF -MAINT	BUTTON-PRESSED	STAFF-DATA in STAFF-MAINT	ROWS-FETCHED
Java Class Name	Staffmn	StaffMaint	NA	StaffMaint _StaffData	NA
Java Object Name	staffmn	staffMaint	buttonPressed	staffData	rowsFetched
Java Package Name	StaffPkg (from /PACKAGENAME)	StaffPkg (from /PACKAGENAME)	NA	StaffPkg (from /PACKAGENAME)	NA
Java Get Method Name	NA	getStaffMaint	getButtonPressed	getStaffData	getRowsFetched
Java Put Method Name	NA	setStaffMaint	setButtonPressed	setStaffData	setRowsFetched

Data Type Mapping

The following table shows the Java data types derived from VisualAge Generator data item definitions.

Table 45. 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
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

Data Format Conversion Considerations

Numeric Conversion Considerations

- VisualAge Generator to Java:
  - 16-18 digit numbers with decimal places are precise to a maximum of 15 digits.
- Java to VisualAge Generator:
  - Java floating point numbers that have more precision than the corresponding VisualAge Generator item are rounded when converted to VisualAge Generator numbers.
  - If a Java number is greater than the maximum for the corresponding VisualAge Generator data item, an exception is raised. The exception may be a java.lang.ArithmeticException or an com.ibm.vgj.cso.CSOException, message CSOE7953, depending on whether the error is detected during marshalling or numeric format conversion.

    The calling method must ensure that numeric values are within the range that can be processed by the VisualAge Generator server program.

Character String Conversion Considerations

- VisualAge Generator to Java:
  - Trailing blanks are truncated when a VisualAge Generator character item is converted to a Java character string.
- Java to VisualAge Generator
  - Java Unicode characters that do not have a corresponding character in the target code page are mapped to the SUB character for the code page. No exception is raised.
  - Java strings are padded with blanks if shorter than the VisualAge Generator data item, and truncated to VisualAge Generator item length, if longer than the VisualAge Generator data item. No exception is raised.

Power Server API Tracing and Debugging Environment Variables

To enable tracing in the Power Server API (rather than just from the Java wrapper support) when called from Java applications, set CSO environment variables from the window in which the application is started. To enable tracing in the Power Server API when called from Java applets, set CSO environment variables from the window in which the Session Manager is started. To enable tracing in the Power Server API when called from servlets or JSPs, set CSO environment variables so that they are picked up by your web application server.

Tracing Errors

To trace only errors, use the following:

```
SET CSOTROPT=1
SET CSOTROUT=trace_file_name
```

The default trace file name is CSOTRACE.OUT in the directory from which the Session Manager or Java application was started.

Tracing Service Calls

To trace all service calls to the PowerServer API, use the following:

```
SET CSOTROPT=2
SET CSOTROUT=trace_file_name
```

The default trace file name is CSOTRACE.OUT in the directory from which the Session Manager or Java application was started.

Tracing Parameter Contents

To trace contents of parameters before and after server calls, use the following command:

```
SET CSO_DUMP_DATA=ALL
```

to trace calls to all programs. Parameter contents are written to file CSODUMP.OUT in the directory from which the Session Manager, web application server, or the Java application was started.

Exception Handling

The server wrapper raises a CSOException when an error is encountered on a call. The catching routine can retrieve the VisualAge Generator middleware CSO error message by calling the CSOException.getMessage() method.

ErrorMessages

CSO7950E

CSO7950E Parameter descriptions and parameters for call to application %1 are incompatible.

Explanation: You are using a generated JavaBeans wrapper class to call a VisualAge Generator application. The wrapper class has the same name as the application. The parameter description generated for the class definition does not match the parameters being passed. This can happen for one of the following reasons:

- Class definitions for the server or record parameters were modified after they were generated.
- An array variable in a record parameter object was reassigned with a different array size.

- The record parameter class was generated separately from the server class after the record definition had changed. This can happen if the same record is used as a parameter in one or more applications.

User Response: Regenerate the server application to rebuild the server and record classes. Recompile the generated Java classes. Ensure that classes calling the server do not reassign array variables with a different size.

CSO7951E

CSO7951E The parameter description for a call to application %1 is invalid. Parameter number is %2. Invalid item description length is %3. Parameter offset is %4.

Explanation: You are using a generated JavaBeans wrapper class to call a VisualAge Generator application. The wrapper class has the same name as the application. During parameter format conversion, the end of the data was reached before all data items where converted, indicating the generated parameter description is invalid. This can happen for one of the following reasons:

- Class definitions for the server or record parameters were modified after they were generated.
- An array variable in a record parameter object was reassigned with a different array size.
- The record parameter class was generated separately from the server class after the record definition had changed. This can happen if the same record is used as a parameter in one or more applications.

User Response: Regenerate the server application to rebuild the server and record classes. Recompile the generated Java classes. Ensure classes that call the server do not reassign array variables with a different size.

CSO7952E

CSO7952E Unknown item type in a parameter description for a call to application %1. Parameter number is %2. Invalid item description length is %3. Parameter offset is %4.

Explanation: You are using a generated JavaBeans wrapper class to call a VisualAge Generator application. The wrapper class has the same name as the application. An unknown item type was encountered in the parameter description for the server application.

This can happen for one of the following reasons:

- Class definitions for the server or record parameters were modified after they were generated.

- An array variable in a record parameter object was reassigned with a different array size.
- The record parameter class was generated separately from the server class after the record definition had changed. This can happen if the same record is used as a parameter in one or more applications.

User Response: Regenerate the server application to rebuild the server and record classes. Recompile the generated Java classes. Ensure classes that call the server do not reassign array variables with a different size.

CSO7953E

CSO7953E Numeric overflow occurred when converting a Java parameter to server data format on a call to application %1. Parameter number is %2. Item type is %3. Item length in bytes is %4. The number of decimal places is %5. Item offset in parameter is %6.

Explanation: You are using a generated JavaBeans wrapper class to call a VisualAge Generator application. The wrapper class has the same name as the application. A number being passed to the server is greater than the largest number that can be stored in the VisualAge Generator data item associated with the parameter. The parameter offset is the hexadecimal offset of the server data item in the parameter structure as defined to the server.

User Response: The client developer must modify the client to check that value being passed is within the range that is acceptable to the server.

CSO7955E

CSO7955E %1, %2

Explanation: The application or applet you are running uses a generated JavaBeans wrapper class to call a VisualAge Generator application. An unexpected Java exception was caught while attempting to call the server.

The message text consists of the name of the Java exception followed by the Java message thrown with the exception.

Possible causes include the following:
- The generated wrapper classes were modified after they were generated.
- Overflow occurred when a floating point parameter or variable contained a value larger than the corresponding server data item can contain.
- VisualAge Generator conversion tables for Java (files with extension .JCT) are not in a directory accessible to the Java application or VisualAge Generator unit of work server for applets.

User Response: Use the Java error description in problem analysis.

If overflow is the problem, modify the client code to ensure that the values being sent to the server are within the range that is acceptable to the server.

CSO7956E

CSO7956E Server unit of work for applet was canceled.

Explanation: You are using an applet that calls generated VisualAge Generator server applications. A VisualAge Generator PowerServer session was started to handle communications between the applet and the server systems. The session was canceled and session resources released because the applet was inactive (did not make any calls to servers) for an extended period of time.

User Response: The applet should be coded to close the server unit of work if the applet user does not make any service requests within a reasonable time.

CSO7957E

CSO7957E Conversion table name %1 is invalid for Java character conversion.

Explanation: You are using a generated JavaBeans wrapper class to call a VisualAge Generator application and specified a conversion table that is not valid for use in converting Java character strings to server format.

The name you specify for a Java conversion table must have the format CSOpcccc where p identifies the server environment type (I = Intel for OS/2 and Windows, X = Unix for AIX and other Unix environments, and E = EBCDIC for MVS, VM, VSE, and OS/400), and cccc is the one- to four-digit code page number for the character set in use on the server environment.

User Response: Specify the conversion table name in the correct format.

Part 4. Calling Server Programs from Non-VisualAge Generator Clients

Chapter 19. Using Interspace to Call Server Programs from Visual Basic, PowerBuilder, or ActiveX Clients

VisualAge Generator and Interspace by Planetworks have teamed up to provide an exciting and powerful new offering for enterprise customers. Using the Interspace development framework and middleware, developers can create Visual Basic or PowerBuilder GUIs that call robust, scalable, VisualAge Generator transaction server programs to access enterprise data.

In this chapter, we describe how to build a Visual Basic client for the VisualAge Generator sample server program STFLIST using Interspace. Included in the Samples directory is a self-extracting file, VBSAMP.exe, containing the sample files that were created using these instructions. To run the executable version of the STAFF project, STFPROJ.exe, you need to have installed VisualBasic version 5.0 or greater and VisualAge Interspace.

Sample files are:

STFLIST.cat
Service and service interface defined by Interspace

STFLIST.esf
STFLIST service code ESF file generated by Interspace

STFAPP.esf
STFAPP ESF file modified from STFLIST example

STFLIST.bas
STFLIST wrapper for Visual Basic program generated by Interspace

STFLISTM.bas
A Visual Basic program to initialize the project

STFPROJ.vbp
Visual Basic project properties

STFLOGIN
A Visual Basic userid/password authentication form

STFGUI
A Visual Basic GUI form

Defining the Server Program Interface

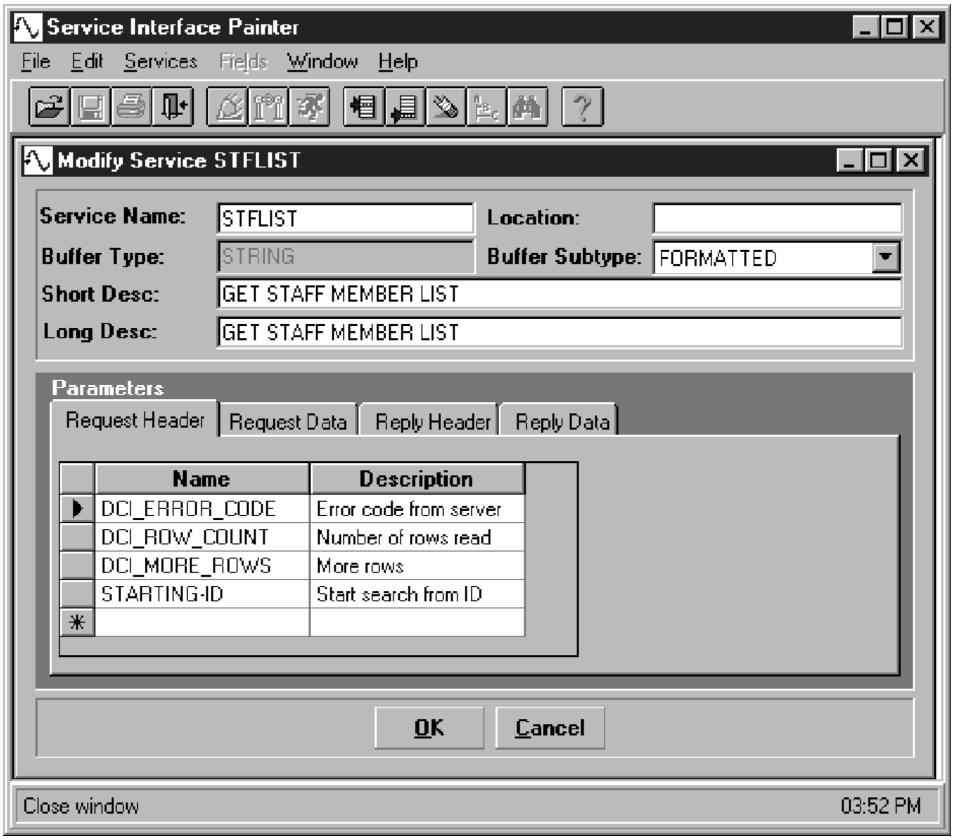
Developing an Interspace-enabled client begins with the Interspace Service Interface Painter. This tool is used to define the data passed between the client and server, to perform test calls to the server, and to generate the objects that call the server program. We used the Interspace Painter to define a service interface for the sample program, STFLIST, which is shipped with VisualAge Generator Developer.

The steps we followed were:

1. Define the fields used in the service interface.

   Fields are equivalent to VisualAge Generator data items. The fields we defined are located in the STFLIST.cat file.

   Note that numbers were defined using type short or integer for integers or decimal for numbers with decimal places.
2. Define the service and service interface. The service is the equivalent of a VisualAge Generator server (remote called batch) program. The service interface is defined as a set of request data flowing to the server, and reply data returned from the server.
3. In Interspace 5.1, you must define the request data and reply data exactly the same for calling VisualAge Generator servers. If repeating data is defined in the interface, the equivalent of an array in VisualAge Generator, you must also define and use Interspace control fields. These fields should be defined only to the request header and should be defined as the first three fields in the request header. The following example illustrates how these Interspace control fields are defined to the Request Header:


Service Interface Painter
File Edit Services Fields Window Help
Modify Service STFLIST
OK
Cancel
Close window
03:52 PM

Service Name: STFLIST
Location:
Buffer Type: STRING
Buffer Subtype: FORMATTED
Short Desc: GET STAFF MEMBER LIST
Long Desc: GET STAFF MEMBER LIST

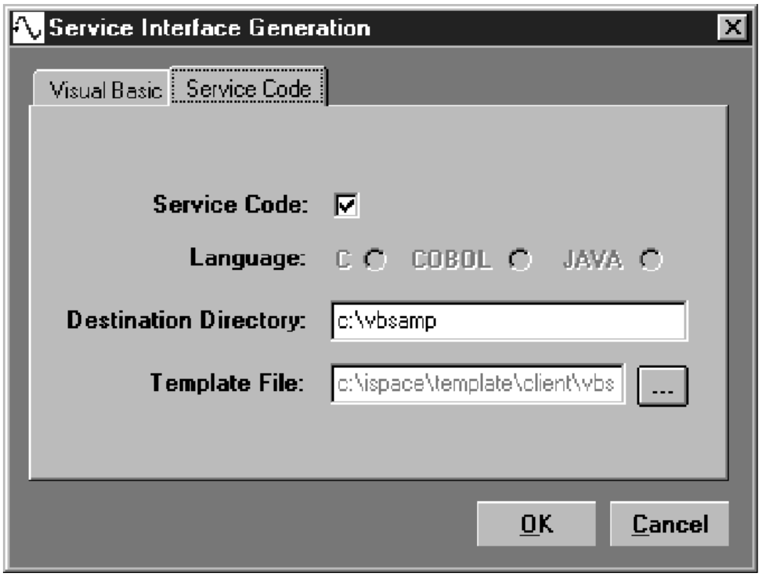
Parameters: Request Header | Request Data | Reply Header | Reply Data

Name	Description
DCI_ERROR_CODE	Error code from server
DCI_ROW_COUNT	Number of rows read
DCI_MORE_ROWS	More rows
STARTING-ID	Start search from ID

Figure 14. Exapmle of the Request Header definition

Notice that the Interspace control fields are not included in the reply header.

4. Use the Generate function from the Service Interface Painter to generate External Source Format for the program and parameter record member for VisualAge Generator. Interspace uses the term *service code* to refer to the ESF file. You can see the generated service code in file STFLIST.ESF.



Service Code: ✓

Language: C ○ COBOL ○ JAVA ○

Destination Directory: c:\vbsamp

Template File: c:\ispace\template\client\vbs

Figure 15. Exapmle of ESF generation for the sample server program

5. Import the external source format file into the VisualAge Generator library and code the remainder of the server program using VisualAge Generator Developer. We took most of the code from the existing STFLIST sample program and reworked it to work with the Interspace generated parameter record. The modified source code is in file STFAPP.ESF.

Testing the Server Program

After you define the service interface for the VisualAge Generator server program server program, you can test the service using the Interspace Service Tester. To simulate the interaction with the server program before a GUI client program has been defined using the Interspace Service Tester:

1. Setup the PowerServer environment so that the VisualAge Generator test facility is started when the Interspace Service Tester calls the server program. When both Interspace Service Tester and VisualAge Generator test facility are on the same Windows NT system, use the following linkage table entry to point to the test facility as the test server:

   ```
   :CALLLINK APPLNAME=* LINKTYPE=REMOTE REMOTECOMTYPE=IPC
             REMOTEAPPTYPE=ITF CONTABLE=PICTURE
   ```

   CONTABLE (the conversion table name) must always be specified for Interspace clients, because numbers must be translated from Interspace format to VisualAge Generator format. Specify the conversion table name as PICTURE whenever your client and server have the same code page.

2. Ensure that PowerServer points to the linkage table file by setting environment variable CSOLINKTBL to the file name. For example,

   ```
   SET CSOLINKTBL=c:\vag\staff.lnk
   ```
3. Start the test session on Interspace, by selecting **View All** from the **Services** menu to open the **Services** window. Select the service name to be tested and select **Test** from **Services** menu. An interactive Interspace Service Tester displays where data values can be entered into the appropriate fields and executed to start the service middleware.

Building Visual Basic GUIs

After the service interface and server program were tested to our satisfaction, we followed the following steps to build the Visual Basic GUI:

1. Generate the Visual Basic wrapper module for the STFLIST service
2. Copy STFLIST module in a Visual Basic project
3. Add modules to the project
4. Build the forms for the GUI
5. Test the Visual Basic program

Generating Visual Basic Functions that Call the Server

Interspace uses the *service definitions* and the *service call template* to generate the appropriate Visual Basic structures and functions. The generated code contains functions that invoke Interspace functions for moving data back and forth between Visual Basic structures and middleware communication buffers. The generated code is written to a Visual Basic file with a *.BAS* extension and a filename equal to the service name.

The *vbsync.tpl* template should be used when generating the service for a VisualAge Generator called server program, since these are synchronous calls. The generator generates several Visual Basic structures and functions that include the name of the service.

Predefined Interspace Functions

The most important function of these generated functions is called *receive_<service>_sync*, where *<service>* is the service name. This function invokes other predefined and generated functions to interface with the Interspace middleware services. These predefined and generated functions make up what is called the GUI-Enabling Layer (GEL), which is the top layer. These predefined functions are contained in DCIGEL.BAS. The Distributed Processing Layer (DPL), the middle layer, contains functions that provide additional middleware services. These functions are contained in dcidpl.dll and are declared in DCIDPL.BAS.

Developing A Visual Basic GUI for the STFLIST Server Program

Before a Visual Basic GUI can call a VisualAge Generator server program using the Interspace function, the Interspace environment must be initialized. The function *dcifx_init()*, which initializes the environment, along with other predefined functions, are contained in DCIGEL.BAS, as mentioned above. When terminating the GUI, the Interspace environment should be cleaned up by using the *dcifx_exit()* function. The sample code shows you where these functions and other Interspace functions should be coded in your Visual Basic application:

```
Public STFLISTrequestdata As FLIST_request_data
Public STFLISTreplydata As STFLIST_reply_data
Public STFLISTrequestmsg As STFLIST_request_msg
Public STFLISTreplymsg As STFLIST_reply_msg

Public pword$   'user password
Public user$    'user name

Public retcode%   'global error code
Public NumofButtons%   'used for the dcifx_show_error

Public Const DistributedEnvironment = "VISGEN" 'environment you are
     'connecting to
Public Const AppToConnectTo = "STFLIST"        'the app
     'connecting to
Public Const Title = "Interspace Reported Error" 'used for the
     'dcifx_show_error

Public Sub Main()
  STFGUI.Show
  stfLogIn.Show 1
End Sub
```

Visual Basic Modules

The steps for developing a Visual Basic application are as follows:

1. Add the modules (DCIGEL.BAS and DCIDPL.BAS) containing the predefined Interspace functions to the Visual Basic project.
2. For each service (server program) that will be called by the GUI, add the module that was generated from Interspace to the Visual Basic project. For our example, the generated module, STFLIST.BAS, was added to the Visual Basic project.

Adding a Main Subroutine

We created another Visual Basic module called STFLISTM.BAS to keep from modifying the Interspace-generated modules in the event that the service interface changes and the service has to be regenerated. In the STFLISTM.BAS module, we defined a Main subroutine for our Visual Basic GUI and additional variables. The script for the Main subroutine is the first code to be executed and is used to control the initial flow of the GUI.

Included in the additional variables section is a copy of all the service interface parameters for each service call. These are the actual parameters that should be used when making the function calls. The naming convention we adopted for naming the variables was to remove the underscore from the original name. For instance, *STFLIST_request_data* was defined with the name *STFLISTrequestdata*.

Coding the Visual Basic Forms

The Visual Basic GUI created for our server program contains two forms: a login form (STFLOGIN.frm) and the main form (STFGUI.frm), which controls the interaction with user.

The function, dcifx_init(), which initializes the Interspace environment, is included in the login form (STFLOGIN.frm).

```
Private Sub getListCmd_Click()
 'Initialization section
 Dim stfname As String * 15
 Dim STFListEntry As String * 50

 'Use STARTING_ID entered by the user
 STFLISTrequestmsg.header.starting_id = starting_id.Text

 'Allocate space in memory for the following arrays
 ReDim STFLISTrequestmsg.data(1) 'very important
 ReDim STFLISTreplymsg.data(1) 'very important

 getListCmd.Enabled = False 'Disable until successful server call

 'The parameters are messages:
 '1.STFLISTrequestmsg - data flowing to the server program
 '2.STFLISTreplymsg - data being returned from the server program
 ' In VB terms: A user defined type. These two were defined in the
 ' STFLIST.BAS file.
 ' Their definition is based on the repository file for this service.
 ' The STFLIST.BAS file was generated using the Interspace painter.

 retcode = receive_STFLIST_sync(STFLISTrequestmsg, STFLISTreplymsg)

  'You always use error handling with an Interspace enabled application
  ' The dcifx_show_error provides information about the error.

 If retcode <> 0 Then 'zero is Interspace success value
  If retcode = -1 Then
  Call dcifx_show_error(NumofButtons, Title) 'Displays Interspace
   'provided error msg
  retcode = 0
  End If
 End If

 getListCmd.Enabled = True 'Call to server was successful
 stfListBox.Clear 'Clear the list box before populating again

 Select Case STFLISTreplymsg.row_count 'using this user defined type to
  'see how many rows were returned by the service

 Case Is > 0                             'At least one row was returned
  For i = 1 To STFLISTreplymsg.row_count 'Number of rows read from the
    'database
   stfname = Format(STFLISTreplymsg.data(i).NAME_WS, "@@@@@@@@@@@@@@@")
   STFListEntry =
Format(Str(STFLISTreplymsg.data(i).STAFFIDX_WS), "@@@@@") +
"| " + stfname +
"| " + Format(Str(STFLISTreplymsg.data(i).SALARY_WS), "@@@@@") + "| " +
     Format(Str(STFLISTreplymsg.data(i).COMM_WS), "@@@@@") + "|"
   stfListBox.AddItem STFListEntry, (i - 1)
  Next i

 Case Is = 0 'No entries found
 End Select

End Sub
```

Testing the Visual Basic Program

Visual Basic provides robust test facilities for running and debugging the Visual Basic program. Use these facilities to set breakpoints and watch points as you are testing and debugging your program.

You have flexibility in how you test your client and server programs together. You can test the Visual Basic program calling the server program in the VisualAge Generator Test Facility. If you are satisfied with the server program, generate the server program, and test it with Visual Basic calling the generated server program. Once the complete client/server application has been thoroughly tested, the Visual Basic program can be compiled into an executable (EXE).

Deploying The Visual Basic Application

After an executable has been created for your Visual Basic application and is ready to be distributed to the end users, it is important that all the necessary files are distributed with the executable and that the middleware runtime components are properly installed on each machine. When distributing the Interspace-enabled Visual Basic application for VisualAge Generator middleware, ensure that all the files in the Interspace runtime subdirectory (x:\ispace\runtime) are distributed to the client machine. This directory should contain the Interspace runtime DLL, DCIDPL.DLL, and the DIL DLL for VisualAge Generator, VGENDIL.DLL.

In addition, you need to distribute the latest version of the Visual Basic runtime DLL. Currently, these DLLs are vb40016.dll and vb40032.dll, depending on whether your application is 16 or 32 bit. If your application uses any OCX objects, you also need to distribute the runtime DLLs for these objects.

The runtime DLLs for Interspace, Visual Basic, and OCX (if it is used) should be located in a directory that is defined in the DOS Path of the client machine. In addition, the Interspace control file, ISPACE.INI, and the repository file containing the service definitions should be copied to the same location.

To complete the setup, you need to install and properly configure VisualAge Generator runtime services and the underlying middleware (CICS Client, Client Access/400, and so on). Refer to the product installation guide for information on the installation and configuration of these products.

You also need a runtime linkage table on the client to specify the location of the server to the PowerServer middleware.

Moving the Server Program to Other Platforms

Our example put the server program on an MVS CICS system. To put the server program on any of the other VisualAge Generator server platforms (for example, IMS, VSE CICS, OS/400, or AIX), regenerate the server program for the new environment and modify the linkage table on the client system to point to the new server. No change is required in the server application code or to the GUI client program.

Using A Different GUI Development Tool

The same server program can be used with PowerBuilder, Java, and ActiveX clients. Use Interspace to generate PowerBuilder DataWindow objects, Java classes, or ActiveX controls that encapsulate the call to the server program, then use the appropriate GUI development tool for the type of object generated. No change is required to the server program.

Generating the Server Program

Once the server program has been verified using the test facility, use VisualAge Generator to generate the server program as a remote called batch application for any of the server environments that are callable from the client system.

When you generate, specify the following options in the generation linkage table specified for the server program:

```
    :CALLLINK APPLNAME=* LINKTYPE=REMOTE PARMFORM=COMMDATA.
```

Building GUIs

Use Service Interface Painter to generate GUI objects for the Visual Basic, PowerBuilder, or ActiveX environments. Select a synchronous template when generating the client objects, since all the calls to VisualAge Generator servers are synchronous. Use an interactive development environment appropriate for the type of generated object to build the GUI client programs that call the server program.

Preparing the Client Environment for Calling the Server

When you have created the client executables, you need to complete the client setup by doing the following:

- Install the Interspace runtime component, VisualAge Generator runtime services, and the underlying middleware (CICS Client, Client Access/400, etc.) on the client systems. Refer to the product installation guide for information on the installation and configuration of these products.
- Create a runtime linkage table on the client to specify the location of the server to the PowerServer middleware. For example, the following is an example of a linkage table entry for calling an MVS CICS server application:

  ```
  :CALLLINK APPLNAME=STAFF* LINKTYPE=REMOTE REMOTECOMTYPE=CICSCLIENT
                 LOCATION=CARMVS1 CONTABLE=ELACNENU
  ```

  CONTABLE (the conversion table name) must always be specified for Interspace clients, because numbers must be translated from Interspace number format to VisualAge Generator number format. Specify the conversion table name as PICTURE whenever your client and server have the same code page. Otherwise, specify the conversion table name (ELACNENU, as in the example, for Windows to MVS conversion) just as you would for a VisualAge Generator client running on the same platform as the Interspace client.
- Ensure that PowerServer points to the linkage table file by setting environment variable CSOLINKTBL to the file name. For example,

  ```
  SET CSOLINKTBL=c:\vag\staff.lnk
  ```

Tracing and Debugging Server Calls

To trace server calls from the Interspace DIL, set CSO trace options from the window in which the application is started. Tracing is done within the PowerServer middleware.

Tracing Communication Errors

To trace errors, use the following:

```
SET CSOTROPT=1
SET CSOTROUT=trace_file_name
```

The default trace file name is CSOTRACE.OUT in the directory where the GUI client is running.

Tracing Service Calls

To trace all service calls to the PowerServer API, use the following:

```
SET CSORROPT=2
SET CSOTROUT=trace_file_name
```

The default trace file name is CSOTRACE.OUT in the directory where the GUI client is running.

Tracing Parameter Contents

To trace contents of parameters before and after server calls, use one of the following examples:

To trace calls to a specific program, use the following:

```
SET CSO_DUMP_DATA=server_program_name
```

To trace calls to all programs, use the following:

```
SET CSO_DUMP_DATA=ALL
```

Parameter contents are written to file CSODUMP.OUT in the directory where the GUI client is running.

Part 5. Distributed Logic Using Asynchronous Processing

Chapter 20. Implementing Client/Server Processing Using the Message Queue Interface

Message queueing provides an alternative to implementing program communication as a remote procedure call from client to server. With message queueing, programs communicate by writing to and reading messages from queues.

The following are benefits of message queueing:
- Communicating programs can run in parallel.
- Communicating programs do not need to be running concurrently.
- Intermittent communication link failures do not prevent messages from being delivered.

For more information on message queueing and the design of message queueing programs, refer to the *VisualAge Generator User's Guide*(SH23-0268-01).

Chapter 21. Implementing Client/Server Processing in CICS Using the CREATX Service Routine

In CICS client programs, you can use the CREATX service routine to start a transaction on a remote system to process a message passed to the transaction. The message is the data portion of the CREATX parameter record. The message data is moved into the working storage area of the started program when the transaction starts.

A CALL CREATX is an asynchronous remote procedure in which the requesting program initiates the request and does not wait for the remote program to complete the request before continuing processing. The remote transaction runs independently of the local transaction and is not associated with any terminal.

Use CREATX when the remote program might take a long time to complete. In this way, the local program can continue with other processing while waiting for the remote program to complete processing. If required, the remote program can return information to the caller by storing information in a remote file accessible to the program that issued the CREATX. The remote program can also start another remote transaction on the system from which the original CREATX was issued.

If the initial program completes its other work before the remote program has updated the response file, the initial program can use the EZEWAIT function to suspend processing for a specified time period. Design your program to return to the program user with an appropriate response if the remote program cannot complete processing in a reasonable time period.

Figure 16 on page 384 shows an overview of a client/server program using the CREATX service routine. After the transaction is started by the CALL CREATX statement, the programs communicate through a distributed file. The remote program stores its response in a distributed file accessible by the original program. "Chapter 23. Accessing Distributed Files in CICS" on page 393 contains information on remote files.

Note: Non-CICS programs cannot use CREATX to start a remote call.

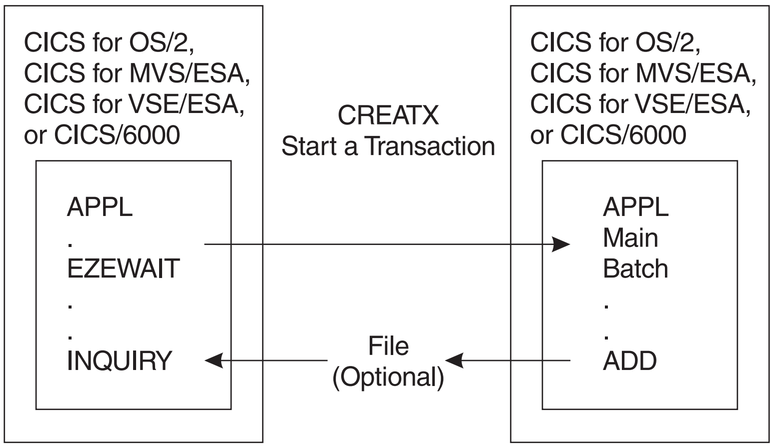


Figure 16. Client/Server Program Using CREATX Service Routine

Defining a Remote Program

Define the remote program as a main batch program. The working storage record defines the structure of the information received by the program when it is started.

If the remote program performs printing, it must move the print destination into EZEDESTP.

Defining the CALL CREATX Statement

The syntax of the CALL CREATX statement for a remote CREATX is the same as that for a local CREATX. However, the parameters PRID and RECIP, specified on the CREATX service routine, are ignored for remote programs.

The record passed on the CREATX call defines the structure of the information passed to the remote program. The information in the record following the length and transaction name in the first 10 bytes is passed to the remote program. The program can set the length field in the CREATX record to control the amount of information passed to the remote program.

Testing CREATX Calls

The two programs are tested separately. The test facility does not emulate the CREATX service; instead, it treats the CREATX service call as a call to a program named CREATX.

To emulate CREATX processing, you can write your own CREATX program that writes the CREATX record to a file.

When you finish testing the initial program, you can run a second program to read the CREATX record from the file. You then use the XFER statement to pass control to the program to be started using the CREATX record (without the first 10 bytes) as the passed XFER working storage record.

Format conversion is not performed when running in the test facility, and the contents of EZELOC and EZECONVT are ignored on CREATX calls. Calls to EZEWAIT are also ignored, except for validating the parameters on the call. Calls to EZECONV should be bypassed when running on the test facility.

Generating Client/Server CREATX Calls

When you generate programs that use CREATX calls, use the linkage table to identify the CREATX records being passed to a remote transaction. The transaction name itself is identified in the record at run time. For example, if program CLIENT is to issue a CREATX passing record REMDATA, specify the following in the linkage table:

```
:crtxlink recdname=remdata  linktype=remote ...
```

The linkage table is required only when you are generating the program issuing the CREATX service routine. Specify the target system according to which the program being generated is to run (OS2CICS, MVSCICS, or VSECICS).

Identifying the Location of the Remote Transaction

If you want the location of the remote transaction to be obtained from the transaction entry at run time, specify the following in the linkage table entry for the CREATX record:

```
:crtxlink recdname=remdata  linktype=remote  location=cics ...
```

There must be a CICS transaction or PCT entry for the remote transaction defined on the CICS system from which the CREATX is issued. Specify the remote system identifier and the remote transaction code (the name that the transaction is known by on the remote system) in the local transaction entry. The transaction must also be defined on each system that the transaction can be created.

If the CICS system identifier (SYSID) value for the location of the remote transaction is loaded by the calling program into EZELOC prior to the CALL CREATX statement, specify the following in the linkage table for the remote transaction:

```
:crtxlink recdname=remdata  linktype=remote  location=EZELOC ...
```

Define the calling program to load the location into EZELOC at run time if you want to start the same remote transaction installed on different CICS systems. The value in EZELOC must be the SYSID for one of the remote systems defined to the local CICS system.

For more information on linkage tables, see "Appendix B. Linkage tables" on page 405.

Converting Data Format on a Call CREATX

If your program is starting a transaction from a mainframe (or from a workstation) and you want data conversion to be performed automatically on a CALL CREATX, specify the CONTABLE keyword in the entry for the CREATX record in the linkage table, as shown in the following example:

```
:crtxlink recdname=remdata  linktype=remote  contable=conversion_table_name ...
```

Data is automatically converted on the calling system before issuing the CICS START command. Conversion is performed based on the data structure defined for the CREATX record. Use automatic conversion only if the record values match the structure definition.

If you want your remote transaction to run on computers that support different code pages, specify EZECONVT instead of a conversion table name in the CRTXLINK entry and have your program move the appropriate conversion table name into EZECONVT before issuing the CALL CREATX.

There are times you might want to control automatic conversion. For example, when your program sets the remote location using EZELOC, you can set conversion off when the remote transaction is on another workstation and then set conversion on when the remote transaction is on a mainframe. To do this, specify EZECONVT as the conversion table in the crtxlink entry. If you do not want the conversion to be performed, move blanks to EZECONVT prior to the CALL CREATX statement. If you do want the conversion to be performed, move a conversion table name to EZECONVT.

The "Appendix C. Converting Between Client and Server Data Formats" on page 433 section contains more information on data format conversion, including a description of the conversion algorithms and a list of conversion tables provided with the product.

Using the EZECONV Service for Redefined CREATX Records

If you do not want automatic conversion performed at the time of the CALL CREATX statement, omit the contable keyword from the crtxlink entry. Do not use automatic conversion if the data structure of the argument can vary from one CREATX statement to another. The program can use the EZECONV service with record redefinitions to perform conversion when record formats vary. When you use a redefined record as the target record of a call to EZECONV, the redefined record is treated as a fixed length record.

If you are using EZECONV instead of automatic conversion on the CREATX, define the data structure for the CREATX record so that the first 10 bytes of the record (the length and transaction name) are not converted. The length and transaction name must be in local system format at the time the CREATX is processed.

Handling Link Failures

The remote CALL CREATX is generated as a CICS START request to a remote transaction. If the START is unsuccessful for any reason, including when the communication link is not available, CICS returns with error information in the CICS EIB. If the REPLY option is specified on the CALL CREATX statement, the program moves one of the following values into EZERT8:

Value	**Meaning**
00000000	Successful CREATX
00000203	Transaction identifier not valid
00000207	System identifier not valid
00000208	Link out of service
ffrrrrrr	Other CICS error where ff is the hexadecimal representation of EIBFN byte 0 and rrrrrr is the hexadecimal representation of EIBRCODE bytes 0–2.

If the REPLY option is not defined on the CALL CREATX, the calling program ends with error messages.

Committing Changes in Remote Transactions

The transaction started using CALL CREATX is a separate logical unit of work. Any resources updated by the remote transaction are committed independently of the program that issued the CALL CREATX statement.

Detecting Errors in Programs Started by CREATX

If runtime services detects an error while the started program is running, the messages describing the error are logged according to the diagnostic control options in effect on the remote system. The starting program is not notified of the error. The starting program should therefore be designed to handle the condition when no response is returned from the started program within a specified time period.

Part 6. Distributed Data

Chapter 22. Accessing Remote Relational Databases

IBM's DB2 database managers implement the Distributed Relational Database Architecture (DRDA) and provide client database access products that allow your programs to access remote databases in the same way they access local databases.

The local database manager or client access product handles communications to the remote database manager and data format conversion in a manner that is transparent to the program developer. Communication failures are communicated to the program using SQL return codes. The remote database updates are included in the program's unit of work and are committed or rolled back in conjunction with the other recoverable resources controlled by the program.

Call the VisualAge Generator EZECONCT service routine during program runtime to dynamically change the current database connection, specifying whether a local or remote database is desired. The default database connection is dependent on the VisualAge Generator program runtime target environment. Refer to the specific target environment runtime manual for the database connection information. The default database can be a remote database. EZECONCT does not support distributed logical units of work, so prior to changing a database connection, the current unit of work must be committed or rolled back.

For more information on developing relational database programs, refer to the developing SQL programs section in the *VisualAge Generator Design Guide* document.

Chapter 23. Accessing Distributed Files in CICS

In client CICS programs, you can access CICS VSAM files and transient data queues at remote locations. File I/O process options provide synchronous access of a remote file. To access remote files, you use I/O process options. The following I/O process options are supported:

- INQUIRY
- SCAN
- SCANBACK
- ADD
- REPLACE
- UPDATE
- DELETE

Note: Non-CICS programs cannot access remote files.

Defining Remote Files

To use remote files, define a serial, relative, or indexed record whose record structure matches the file record structure. The record structure for the file controls how the record data format is converted during automatic data conversion. Use variable length record files wherever appropriate to insure that only the information required is passed on the communication link when a record is accessed.

Defining the I/O Process Options

The syntax for defining process options for remote files is the same as for local files.

Testing the I/O Process Options

When running under the test facility all file I/O operations are performed as local operations. Format conversion is not performed and the contents of EZELOC and EZECONVT are ignored. Calls to EZECONV should be bypassed when running process options under the test facility.

Generating I/O Process Options to Remote Files

When you generate programs that use CICS distributed I/O function calls, identify the file name associated with a record as a remote file in the linkage table. For example, if program CLIENT uses record CREDIT in file HCFILE, create the following linkage table entry:

```
:filelink filename=hcfile  linktype=remote ...
```

Specify the linkage table when generating the program that accesses the file. Specify the target system according to where the program being generated is to run (OS2CICS for OS/2, MVSCICS for MVS, VSECICS for VSE, or AIX6000 for CICS for AIX).

Identifying the Location of the Remote File

If you want the location of the remote file to be obtained from the file entry or DCT entry at run time, specify the following in the linkage table entry for the file:

```
:filelink filename=hcfile  linktype=remote  location=cics ...
```

For a VSAM file, there must be a CICS file or FCT entry for the remote file defined on the CICS system from which the program is running. Specify the remote system identifier and the remote file name (the name the file is known by on the remote system) in the local file entry. The file must also be defined on the system where the file is located.

For a TRANSIENT file, there must be a CICS destination control table (DCT) entry for the remote file defined on the CICS system from which the program is running. Specify the facility type as remote (R), the remote system identifier, and the remote destination name (the name that the destination is known by on the remote system) in the DCT entry. The file must also be defined with a DCT entry on the system where the file is located.

If the SYSID value for the location of the remote file is loaded by the program into EZELOC prior to the I/O process option, specify the following in the linkage table for the remote file:

```
:filelink filename=hcfile  linktype=remote  location=ezeloc ...
```

Define the program to load the location into EZELOC at run time, if you want to access the same file installed on different CICS systems (for example, each site has a local parts file that controls the parts inventory at that site). The value in EZELOC must be the system identifier for one of the remote systems defined to local CICS system.

For more information on linkages tables, see “Appendix B. Linkage tables” on page 405.

Converting Data Format when Accessing a Remote File

If your program uses a mainframe file from the workstation, or a workstation file from a mainframe and you want data conversion to be performed automatically on the I/O process option, specify the CONTABLE keyword in the entry for the file in the linkage table, as shown in the following example:

```
:filelink filename=HCFILE  linktype=REMOTE  contable=conversion_table_name ...
```

Each I/O operation performed by the program automatically converts the data.

- Conversion occurs before the I/O operation when converting from local to remote format for records written to the remote file and for keys used to read the remote file.
- Conversion occurs after the I/O operation when converting from remote to local format for records read from the remote file.
- Conversion is performed based on the data structure defined for the file record structure. Use automatic conversion only if the file record structure matches the structure of each record in the file.

If you want your program to run on computers that support different code pages, specify EZECONVT instead of a conversion table name in the FILELINK entry and have your program move the appropriate conversion table name into EZECONVT before issuing the I/O process option.

There are times you might want to control automatic conversion. For example, when your program specifies a remote location using EZELOC, you can set conversion off when the file is on another workstation and then set conversion on when the file is on a mainframe.

To do this:

- Specify EZECONVT as the conversion table in the FILELINK entry.
- When you do not want conversion performed, move blanks to EZECONVT in your program before an I/O process option.
- When you want conversion performed, move a conversion table name to EZECONVT before an I/O process option.

See “Appendix C. Converting Between Client and Server Data Formats” on page 433 for more information on data format conversion, including a description of the conversion algorithms and a list of conversion tables provided with the product.

Using the EZECONV Service for Redefined Parameters

If you never want automatic conversion performed at the time of the I/O process option, omit the CONTABLE keyword from the FILELINK entry. Do not use automatic conversion if the data structure of the record can vary from one I/O process option to another, as is the case when the file contains redefined records.

The program can use the EZECONV service with the record redefinitions to perform conversion when record formats vary. When using EZECONV with files, remember to convert the record before the process option on write requests and after the process option on read requests. For indexed records, you must also convert the record key item before a read request is issued. When you use a redefined record as the target record of a call to EZECONV, the redefined record is treated as a fixed length record.

If the record is variable length and you are using EZECONV instead of automatic conversion on the file I/O request, the record length or number of occurrences item must be in local format at the time of the I/O operation. To ensure this, either define the record length item or number of occurrences item with the data type of PACK so that the length item has the same format on the host or the workstation, or define the record length outside of the record structure.

Maintaining Position in Remote Files

Have your program issue a SET record SCAN to reset file position prior to any SCAN process if the following conditions are true:

- A remote file is located on a system where the file key has a different format.
- You have specified that the generated program perform automatic data format conversion.
- The previous process for the same file was not a SCAN.

Do this even when CICS is performing key conversion for the file.

Handling Link Failures

The remote I/O process option is generated as a CICS distributed file I/O request. If the I/O request does not complete successfully for any reason, including when the communication link is not available, CICS returns with error information in the EIB. If you specify an error routine and EZEFEC is set to 1 when the process option runs and the /SYSCODES (system return codes in EZERT8) COBOL generation option was specified, the program moves the EIB information to EZERT8 as follows and continues running following an I/O error:

Bytes	Description
0–1	Hexadecimal representation of EIBFN byte 0
2–7	Hexadecimal representation of EIBRCODE bytes 0–2.

If the I/O operation ends with a SYSIDERR (problem accessing the remote file), one of the following values is moved to the first 6 characters of EZERT8:

Value	Meaning
06D004	Name not that of system entry
06D008	Link out of service
06D00C	Name unknown to CICS.

If an error routine is not specified or EZEFEC is not set to 1 when an I/O error occurs, the program ends with error messages.

You should not use an error routine with EZEFEC set to 1 and the /NOSYSCODES generation option.

Committing Changes to Remote Files

EZECOMIT and EZEROLLB requests result in CICS SYNCPOINT requests from the generated COBOL program. If the remote files are defined as CICS recoverable resources, CICS honors the SYNCPOINT request for the remote files as well as local files. Links involving CICS OS/2 or CICS for AIX systems incorporate synchronization level 1 (one phase commit) support; links involving only MVS CICS or VSE CICS systems use synchronization level 2 (two-phase commit) support.

Handling Errors

Error handling for remote files works in the same way as error handling for local files.

Part 7. Appendixes

Appendix A. Java properties

The following sections list the Java properties that are applicable to client/server setup. For a more comprehensive list of properties that are used with VisualAge Generator programs, see the *VisualAge Generator Generation Guide*.

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
```

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:

```
tcpiplistener.trace.file=c:\\temp\\uitrace.txt
```

Appendix B. 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:

```
►►──:calllink──applname=program name──┬──────────────────────┬──────────────────►
                                      │          ┌─32──┐     │
                                      └─bitmode=─┴─16──┴─────┘
```

```
►─┬──────────────────────┬──┬──────────────────────────┬──────────────────────►
  └─library=library name─┘  └─linktype=─┬─DYNAMIC────┬─┘
                                        ├─STATIC─────┤
                                        ├─CICSLINK───┤
                                        ├─REMOTE─────┤
                                        ├─CSOCALL────┤
                                        └─SESSIONEJB─┘

►─┬─────────────────────────┬──┬───────────────────────────┬──────────────────►
  └─externalname=applname───┘  └─parmform=─┬─OSLINK─────┬─┘
                                           ├─COMMPTR────┤
                                           ├─COMMDATA───┤
                                           └─CICSOSLINK─┘

►─┬───────────────────────────────────┬──┬──────────────────────────────┬─────►
  └─remotecomtype=─┬─APPCIMS────┬─────┘  └─remoteapptype=─┬─VG─────┬────┘
                   ├─CA400──────┤                         ├─NONVG──┤
                   ├─CICSCLIENT─┤                         ├─ITF────┤
                   ├─DCE────────┤                         └─VGJAVA─┘
                   ├─DCESECURE──┤
                   ├─DIRECT─────┤
                   ├─EXCI───────┤
                   ├─IPC────────┤
                   ├─JAVA400────┤
                   ├─LU2────────┤
                   └─TCPIP──────┘

►─┬──────────────────────────────────────────┬──┬──────────────────────────┬──►
  └─contable=─┬─conversion table name─┬──────┘  └─location=─┬─EZELOC──────┬┘
              ├─'*'───────────────────┤                     └─system name─┘
              ├─EZECONVT──────────────┤
              ├─NONE──────────────────┤
              └─BINARY────────────────┘

►─┬──────────────────────────────────┬──┬────────────────────────────┬────────►
  └─serverid=──server identifier─────┘  │               ┌─CLIENT─┐   │
                                        └─luwcontrol=───┴─SERVER─┴───┘

►─┬─────────────────────────────────┬──┬───────────────────────┬──┬──────────────────┬──►
  │              ┌─GENERATION─┐     │  │         ┌─INTEL─┐     │  └─providerURL=URL──┘
  └─remotebind=──┴─RUNTIME────┴─────┘  └─binform=┴─HOST──┴─────┘

►─┬──────────────────────┬──┬───┬─────────────────────────────────────────────►◄
  └─package=packagename──┘  └─.─┘
```

Definitions for CALLLINK

applname

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 46:

Table 46. 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 CICSLINK 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 "Advantages of IPC and DIRECT Protocols" on page 41.
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 "Java Support for OS/390 Unix Systems" on page 342.
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 "Advantages of IPC and DIRECT Protocols" on page 43.
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 "Java Support for AS/400 Servers" on page 347.
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 "Conversion Table by Language and Platform" on page 436.

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 EBCIDIC character conversion, as well as binary numeric

conversion. For additional information, see “Conversion Table by Language and Platform” on page 436.

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 “Conversion Table by Language and Platform” on page 436. If EZERNLS is not specified, the default code is ENU.

On MVS or VSE systems, the default conversion table is ELA*xxxxx* 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: • 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
TCPIP	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
TCPIP	Service name as defined in the TCP/IP services file. If the call is from a Java program (GUI or server), the serverid 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.

Table 2 on page 12 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 47. 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 48. Valid Parameters and Linkages for Non-CICS Host Programs

Link Type	COMMPTR	COMMDATA	OSLINK	CICSOSLINK
DYNAMIC			Default	
STATIC			Valid	
CICSLINK				
REMOTE				
CSOCALL				

Table 49. 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 50. 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 51. 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 52

Table 52. Nonstandard Linkages Supported by CALL

Function	Environment	Statement
Call to a PL/I program	MVS/VSE/VM Non-CICS	:calllink applname=*program-name* linktype=static
Call to a COBOL program or program that calls a PL/I program	MVS/VSE/VM Non-CICS	:calllink applname=*program-name* linktype=static
Static COBOL calls	CICS, MVS/VSE/VM Non-CICS	:calllink applname=*program-name* linktype=static ...

Table 52. Nonstandard Linkages Supported by CALL (continued)

Function	Environment	Statement
Dynamic COBOL calls	CICS	:calllink applname=*program-name* linktype=dynamic ...
Pass parameter values instead of pointers in the COMMAREA	CICS	:calllink applname=*program-name* parmform=COMMDATA ...
Call to a DLL in 16-bit mode	CICS for OS/2	:calllink applname=*program-name* linktype=dynamic
Call to a DLL in 16-bit mode	GUI program or test facility	:calllink applname=*program-name* linktype=dynamic bitmode=16 library=*value*...
Call to a DLL (OS/2 or Windows) in 32-bit mode	GUI or C++ programs, or test facility	:calllink applname=*program-name* linktype=dynamic library=*value*...
Call to a shared library (AIX, HP-UX, Solaris) in 32-bit mode	C++ programs	:calllink applname=*program-name* linktype=dynamic library=*value*...
Call to a DLL with a program name longer than eight characters	C++ program	:calllink applname=*program-name* linktype=dynamic [externalname=*entry-point name*]...
Call a CICS for OS/2 program	Test facility	:calllink applname=*program-name* linktype=cicslink parmform=commptr [binform=*value*]
Call a CICS server program on a remote system	CICS	:calllink applname=*program-name* linktype=remote [contable=*value*] [location=*value*] [luwcontrol=*value*] [serverid=*value*]

Table 52. 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=*program-name* linktype=remote [externalname=*value*] [library=*value*] [luwcontrol=*value*] [remoteapptype=*value*] [remotecomtype=*value*] [serverid=*value*]

Call a non-VisualAge Generator MVS or VSE CICS program	Test facility	:calllink
		applname=*program-name*
		linktype=cicslink
		parmform=commdata
		contable=*value*

Call any program from a Java client	Needed when calls are made from one platform to another. It is recommended in all cases.	contable=*CSOpxxxx*

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:

```
►►──:crtxlink──recdname=record name──┬─────────────────────────┬──────────────►
                                     │              ┌─LOCAL──┐ │
                                     └─linktype=────┴─REMOTE─┴─┘

►─┬─────────────────────────┬──┬─────────────────────────────────────────┬──►
  └─package=────packagename─┘  └─contable=──┬─conversion table name─┬─┘
                                            ├─*─────────────────────┤
                                            ├─EZECONVT──────────────┤
                                            └─BINARY────────────────┘
```

```
►─┬───────────────────────┬─┬───┬────────────────────────────────────────────►◄
  │            ┌─CICS───┐ │ └─.─┘
  └─location=──┴─EZELOC─┴─┘
```

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:

```
►►──:dxfrlink──fromappl=from-application name──toappl=to-application name──────────►

►─┬──────────────────────────┬─.──────────────────────────────────────────────────►◄
  │             ┌─DYNAMIC─┐  │
  └─linktype=───┼─STATIC──┼──┘
                └─NONCSP──┘
```

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 53 statements, and which linkage table tags and keywords are needed to control each linkage.

Table 53. 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=*fromhyphen.program-name* toappl=*to-program-name* 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=*from-program-name* toappl=*to-program-name* 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:

```
►►──:filelink──filename=file name──┬──────────────────────────────┬──────────────►
                                   │              ┌─LOCAL──┐      │
                                   └─linktype=────┴─REMOTE─┴──────┘

►──┬─────────────────────────────────────┬──┬────────────────────────────┬──┬─────┬──────►◄
   └─contable=──┬─conversion table name─┬┘  │             ┌─CICS───┐     │  └─.───┘
                ├─*─────────────────────┤   └─location=───┴─EZELOC─┴─────┘
                ├─EZECONVT──────────────┤
                └─BINARY────────────────┘
```

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=ncpgm 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.

Appendix C. Converting Between Client and Server Data Formats

Because of the differences in character and numeric data formats between environments, your program might have to convert data as it is passed between client and servers. In VisualAge Generator programs, you can convert data based on the data item definition of the parameters or record structures as defined to the program.

The type of conversion required is defined using a VisualAge Generator conversion table. The developer specifies whether to do conversion and which conversion table to use for a client/server interaction in the linkage table.

Implementation of conversion tables varies with the environment. On OS/400 systems, VisualAge Generator uses system provided conversion functions for code page conversion. The conversion table consists of a single record file which identifies the code page and system type for both the local client and server systems. On OS/2, AIX, HP-UX, Solaris, Windows, MVS, and VSE systems, the conversion tables are load modules that contain code page translation tables which VisualAge Generator uses to perform character data conversion.

VisualAge Generator provides conversion tables for many national languages. You can define additional conversion tables to support other languages or other code pages.

Specify automatic conversion on a client/server function only when the structure of the information being passed always matches the data item structure of the argument or record as defined in the server program. Do not specify automatic conversion in the linkage table if an argument or file record can have multiple definitions (for example, if redefined records are required for a file). Instead use the EZECONV special function word with record redefinitions to convert the data.

For more information on linkage tables, see “Appendix B. Linkage tables” on page 405.

Conversion Algorithm

Data format conversion is performed on the lowest level items (items with no substructure) in an argument or record definition.

Character (CHA, DBCS, or MIX) data is translated using code page conversion tables.

On EBCDIC-to-ASCII conversion for MIX data, the conversion routine deletes shift-out/shift-in (SO/SI) characters and inserts an equivalent number of blanks at the end of the item. On ASCII-to-EBCDIC data conversion, the conversion routine inserts SO/SI characters around DBCS strings and truncates the value at the last valid character that can fit in the field. If the MIX field is in a variable length record and the current record end is in the MIX field, the record length is adjusted to reflect the insertion or deletion of SO/SI characters. The record length indicates where the current record ends.

For binary (BIN) items, the conversion routine reverses the byte order of the item if either the client or server system uses Intel binary format and the other system does not.

For NUM and NUMC items, the conversion routine converts all but the last byte using the CHA algorithm. The sign half -byte (the first half byte of the last byte in the field) is converted as shown in Table 54:

Table 54. Data Conversion for NUM and NUMC Items

EBCDIC Sign - NUM	EBCDIC Sign - NUMC	ASCII Sign
X'F'	X'C'	X'3'
X'D'	X'D'	X'7'

No conversion is performed for PACK, PACF, HEX, Unicode and filler items.

Avoiding Data Format Conversion Problems

Windows Clients and OS/2, AIX, HP-UX, and Solaris Servers

Windows systems use different ASCII code pages than OS/2 , AIX, HP-UX, and Solaris systems, so ASCII-to-ASCII code page conversion must be used when going from a Windows client to an OS/2 or AIX server. Use the conversion tables shown in Table 57 on page 439 to perform the proper conversion for the national languages shown in the table. Define a custom conversion table for code pages not supported with default tables.

Default Tables for Test and Run-time Data Conversion

You can indicate to VisualAge Generator the data format conversion required by specifying a conversion table name in the linkage table entry associated with the function requiring code conversion, in the EZECONVT special function word, or on an EZECONV service call. The conversion table indicates the code pages and binary data formats in use on the local and remote systems.

Note: Java GUI clients require a different set of conversion table names than other clients. See Table 55 on page 436 for additional information. Default conversion tables are not available for Java GUI clients.

If a default conversion table is selected (conversion table name specified as *), the default table name is determined as in the following way:

- On host systems (MVS and VSE), the default conversion table is EZECNxxx, where xxx is the national language associated with the generated program.
- On workstations, the default conversion table is determined in the following way:
  - The contents of the EZERCVT environment variable is the default table name, if set.
  - If EZERCVT is not set, then the default conversion table name is ELACNxxx, where xxx is the value of the EZERNLS environment variable. The default value for EZERNLS is ENU.

Note: Default conversion tables perform both ASCII/EBCDIC character conversion and binary data conversion.

To perform conversion for code pages other than the default conversion table, specify the name of a conversion table representing the alternate code pages.

The language conversion tables provided with the product are listed below. The code page numbers are those specified in the *Character Data Representation Architecture Reference and Registry*, SC09-2190. The registry identifies the coded character sets supported by the conversion tables.

Default Tables for Generation

VisualAge Generator converts character strings to run-time system format in generated objects that contain mixed character and binary data. Objects that require this conversion are:
- Map group format modules
- Tables

The default conversion table used is based on the generation system (OS/2 or Windows NT), the target run-time environment, and the locale (the location selected in your system setup) specified when the generator is running. The default tables are specified in the file hptrules.nls in the \nls subdirectory. You might modify this file to change the default conversion tables used in your installation.

Conversion Table by Language and Platform

Java applications use Unicode 16-bit character encoding at run time. Java specific conversion tables support conversion of the Unicode text to the appropriate ASCII or EBCDIC code page used on the server program system.

Table 55 lists the conversion tables shipped with VisualAge for Java and identifies the code pages used with the conversion tables for Java programs. The table applies to VCE parts that have 4GL parts on the free form surface, Java parts produced when generating a UI record or Web application, clients using Java wrappers, and all Java clients that are calling server programs. Select the appropriate conversion table based on the system on which the VisualAge Generator server program is running.

Table 55. Conversion table names by language and server system

Language	AIX Server	OS/2 Server	Windows Server	MVS, VSE, or OS/400 Server
Arabic	CSOX1046	CSOI864	CSOI1256	CSOE420
Chinese, simplified	CSOX1381	CSOI1381	CSOI1381	CSOE935
Chinese, traditional	CSOX950	CSOI950	CSOI950	CSOE937
Cyrillic	CSOX866	CSOI866	CSOI1251	CSOE1025
Danish	CSOX850	CSOI850	CSOI850	CSOE277
Eastern European	CSOX852	CSOI852	CSOI1250	CSOE870
English (UK)	CSOX850	CSOI437	CSOI1252	CSOE285
English (US)	CSOX850	CSOI437	CSOI1252	CSOE037
French	CSOX850	CSOI850	CSOI1252	CSOE297
German	CSOX850	CSOI850	CSOI1252	CSOE273
Greek	CSOX813	CSOI869	CSOI1253	CSOE875
Hebrew	CSOX856	CSOI862	CSOI1255	CSOE424

Table 55. Conversion table names by language and server system (continued)

Language	AIX Server	OS/2 Server	Windows Server	MVS, VSE, or OS/400 Server
Japanese	CSOX942	CSOI942	CSOI942	CSOE930 (Katakana SBCS), CSOE939 (Latin SBCS)
Korean	CSOX949	CSOI949	CSOI949	CSOE933
Portuguese	CSOX850	CSOI850	CSOI1252	CSOE037
Spanish	CSOX850	CSOI850	CSOI1252	CSOE284
Swedish	CSOX850	CSOI850	CSOI1252	CSOE278
Swiss German	CSOX850	CSOI850	CSOI1252	CSOE500
Turkish	CSOX920	CSOI857	CSOI1254	CSOE1026

The conversion table names have the format CSOsxxxx, where:

- s = server system type

I	Intel (OS/2, Windows NT, Windows 95)
X	Unix workstation (AIX)
E	EBCDIC (MVS, VSE, VM, OS/400)

- xxxx = server codepage number

Do NOT specify contable="*" in linkage tables entries used when calling server programs from Java. If you do not specify the contable attribute in the linkage table entry for a server program called from Java, the default conversion tables used are CSOI437 for OS/2, CSOI1252 for Windows, CSOX850 for UNIX platforms, and CSOE037 for EBCDIC platforms.

Table 56 on page 438 lists the conversion tables shipped with VisualAge Generator common services and identifies the code pages used with the conversion tables for COBOL and C++ programs:

Note: In the column headers, the "local system" is the system on which the operation requiring code point conversion (client calling server, an EZECONV call, generation, or file migration) is being performed.

Table 56. Conversion Table by Language and Platform

Language	Local system: Any ASCII, remote system: EBCDIC	Local system: Windows, remote system: OS/2	Local system: Windows, remote system: AIX, HP-UX or Solaris	Local system: OS/2, remote system: AIX, HP-UX or Solaris	Local system: OS/2, remote system: Windows	Local system: AIX, HP-UX or Solaris, remote system: OS/2
Arabic	ELACNARA	ELAO2ARA	ELAAXARA	ELAAXARA	ELAWIARA	ELAO2ARA
Brazilian Portuguese	ELACNPTB	ELAO2850	ELAAX850	BINARY	ELAWI850	BINARY
Chinese, Simplified (IBM GB)	ELACNCHS	None	BINARY	BINARY	None	BINARY
Chinese, Simplified (IBM GBK - OS/2 and AIX or HP-UX only)	ELACNGBK	Not supported	Not supported	BINARY	Not supported	BINARY
Chinese, Traditional	ELACNCHT	None	BINARY	BINARY	None	BINARY
Danish	ELACNDKN	ELAO2850	ELAAX850	BINARY	ELAWI850	BINARY
Eastern European, Latin-2	ELACN870	ELAO2852	ELAAX912	ELAAX912	ELAWI852	ELAO2852
English, USA	ELACNENU	ELAO2437	ELAAX437	BINARY	ELAWI437	BINARY
English, United Kingdom	ELACN285	ELAO2850	ELAAX850	BINARY	ELAWI850	BINARY
Finnish	ELACNENU	ELAO2850	ELAAX850	BINARY	ELAWI850	BINARY
French	ELACNFRA	ELAO2850	ELAAX850	BINARY	ELAWI850	BINARY
German	ELACNDEU	ELAO2850	ELAAX850	BINARY	ELAWI850	BINARY
Greek	ELACNGRE	ELAO2GRE	ELAAXGRE	ELAAXGRE	ELAWIGRE	ELAO2GRE
Hebrew	ELACNHEB	ELAO2HEB	ELAAXHEB	ELAAXHEB	ELAWIHEB	ELAO2HEB
Italian	ELACNITA	ELAO2850	ELAAX850	BINARY	ELAWI850	BINARY
Korean	ELACNKOR	None	BINARY	BINARY	None	BINARY
Japanese, Katakana SBCS	ELACNJPN	None	BINARY	BINARY	None	BINARY
Japanese, Latin SBCS	ELACNJPL	None	BINARY	BINARY	None	BINARY
Norwegian	ELACNDKN	ELAO2850	ELAAX850	BINARY	ELAWI850	BINARY
Russian	ELACNCYR	ELAO2CYR	ELAAXCYR	ELAAXCYR	ELAWICYR	ELAO2CYR
Spanish	ELACNESP	ELAO2850	ELAAX850	BINARY	ELAWI850	BINARY
Swedish	ELACNSWE	ELAO2850	ELAAX850	BINARY	ELAWI850	BINARY
Swiss German	ELACNDES	ELAO2850	ELAAX850	BINARY	ELAWI850	BINARY

Table 56. Conversion Table by Language and Platform (continued)

Language	Local system: Any ASCII, remote system: EBCDIC	Local system: Windows, remote system: OS/2	Local system: Windows, remote system: AIX, HP-UX or Solaris	Local system: OS/2, remote system: AIX, HP-UX or Solaris	Local system: OS/2, remote system: Windows	Local system: AIX, HP-UX or Solaris, remote system: OS/2
Turkish	ELACNTUR	ELAO2TUR	ELAAXTUR	ELAAXTUR	ELAWITUR	ELAO2TUR

SBCS Languages Code Page by Platform

Table 57 lists the conversion tables shipped with VisualAge Generator common services and identify the code pages for SBCS languages:

Table 57. SBCS Languages Code Page by Platform

Language	EBCDIC Code Page	Windows Code Page	OS/2 Code Page	AIX or HP-UX Code Page
Arabic	420	1256	864	1089
Brazilian	037	1252	850	850
Danish	277	1252	850	850
Eastern European, Latin-2	870	1250	852	912
English, United Kingdom	285	1252	850	850
English, USA	037	1252	437	437
French	297	1252	850	850
Finnish	298	1252	850	850
German	273	1252	850	850
Greek	875	1253	869	813
Hebrew	424	1255	862	856
Italian	280	1252	850	850
Norwegian	277	1252	850	850
Russian	1025	1251	866	915
Spanish	284	1252	850	850
Swedish	278	1252	850	850
Swiss German	500	1252	850	850

Table 57. SBCS Languages Code Page by Platform (continued)

Language	EBCDIC Code Page	Windows Code Page	OS/2 Code Page	AIX or HP-UX Code Page
Turkish	1026	1254	857	920

Note:
These are the code pages used by the conversion tables listed in Table 56 on page 438.

DBCS Languages Code Page by Platform

Table 58 lists the conversion tables shipped with VisualAge Generator communication services and identify the code pages for DBCS languages:

Table 58. DBCS Languages Code Page by Platform

Language	ASCII CCSID	ASCII SBCS Code Page	ASCII DBCS Code Page	EBCDIC CCSID	EBCDIC SBCS Code Page	ASCII DBCS Code Page
Chinese, Simplified (IBM GB)	1381	1115	1380	935	836	837
Chinese, Simplified (IBM GBK)	1386	1114	1385	935	836	837
Chinese, Traditional (IBM BIG-5)	950	1114	947	937	037	835
Japanese, Katakana SBCS, on OS/2 and AIX or HP-UX systems	942	1041	301	930	290	300
Japanese, Latin SBCS, on OS/2 and AIX or HP-UX systems	942	1041	301	939	1027	300
Japanese, Katakana SBCS, on Windows systems	932	897	301	930	290	300
Japanese, Latin SBCS, on Windows systems	932	897	301	939	1027	300
Korean	949	1088	951	933	833	834

Table 58. DBCS Languages Code Page by Platform (continued)

Language	ASCII CCSID	ASCII SBCS Code Page	ASCII DBCS Code Page	EBCDIC CCSID	EBCDIC SBCS Code Page	ASCII DBCS Code Page
Note: These are the code pages used by the conversion table in Table 56 on page 438.						

Defining Custom Conversion Tables

You can define your own code page conversion tables for languages not directly supported by VisualAge Generator.

Defining Conversion Table Files for OS/2 and AIX Systems

On OS/2 or AIX systems, the conversion table is a file with a single record. The file extension of OS/2 and AIX is CTB. The file identifies the characteristics of the local and remote systems, including the code pages used on the systems. Modify the appropriate fields as required.

Use the HPTBLDCF command to create the conversion table file.

To create a file for a single-byte language, enter the following:

```
HPTBLDCF tname lt rt lsbcscp rsbcscp
```

To create a file for a double-byte language, enter the following:

```
HPTBLDCF tname lt rt lsbcscp rsbcscp ldbcscp rdbcscp
```

The command’s parameters mean the following:

tname Output file name: tname.CTB

lt Local system type:

AI	ASCII Intel (Windows, OS/2)
AX	ASCII Unix (AIX or HP-UX)

rt Remote system type:

AI	ASCII Intel (Windows, OS/2)
AX	ASCII Unix (AIX or HP-UX)
E	EBCDIC (MVS, VSE, OS/400)

lsbcscp
Local system single-byte code page number in decimal

rsbcscp
Remote system single-byte code page number in decimal

ldbcscp
Local system double-byte code page number in decimal

rdbcsdb
Remote system double-byte code page number in decimal

Specify only the first five parameters if creating conversion tables for a single-byte language.

An example of the command used for creating the English conversion table is the following:

```
HPTBLDCF ELACNENU AI E 437 37
```

An example of the command used for creating the Japanese conversion table is the following:

```
HPTBLDCF ELACNJPN AI E 1041 290 301 300
```

Defining Conversion Table Modules for MVS, VSE, and Windows Systems

Use the HPTBLDCA utility to convert conversion table source to assembler source files for MVS and VSE systems. Use the HPTBLDCC utility to convert conversion table source to C source files for Windows systems.

SBCS Conversion Table Source Format

Specify the source for a single-byte conversion table as a 256-byte binary file with the file extension SBC. The conversion table is used as a translation table at run time. The code point for each character to be converted is used as an index into the table and is converted to the code point found at that position in the table.

Two files are required, one for conversion from the local code page to the remote code page, and the other for conversion from the remote code page to the local code page.

DBCS Conversion Table Source Format

Specify the source for a double-byte conversion table as a file with extension DBC. The file consists of fixed length records of 9 characters. Each record’s format is as follows:

Bytes	Description
1–4	Hexadecimal representation of code point of character to be converted
5	Blank
6–9	Hexadecimal representation of output code that the character is converted to

One record is required for each valid double-byte code point in the source code page. The records should be ordered in ascending hexadecimal sequence.

An example of records from a DBCS table is as follows:

```
8FA1 D541
8FA2 D542
8FA3 D543
8FA4 D544
```

Two files are required, one for conversion from the local code pages to the remote code pages, and the other for conversion from the remote code pages to the local code pages.

Using the Build Conversion Table Utility

To convert the source files to assembler format, make the directory containing the sources files the current directory. Then to create conversion tables for a single-byte language, enter the following:

```
HPTBLDCA tname lt rt lsbcs rsbcs
HPTBLDCC tname lt rt lsbcs rsbcs
```

To create a conversion table for a language that uses mixed single- and double-byte characters, enter the following command:

```
HPTBLDCA tname lt rt lsbcs rsbcs lsbcs rdbcs ltorsub rtolsub svx evx
HPTBLDCC tname lt rt lsbcs rsbcs lsbcs rdbcs ltorsub rtolsub svx evx
```

The command's parameters mean the following:

tname Output file name. For HPTBLDCA, the output file is an MVS or VSE assembler source with the name tname.370. For HPTBLDCC, the output file is a windows compatible C file with the name tname.C.

lt Local (client) system type:

AI	ASCII Intel (Windows, OS/2)
AX	ASCII Unix (AIX or HP-UX)

st Remote (server) system type:

AI	ASCII Intel (Windows, OS/2)
AX	ASCII Unix (AIX or HP-UX)
E	EBCDIC (MVS, VSE, OS/400)

lsbcs Input file name: local-to-remote single-byte conversion table. File extension is assumed to be SBC.

rsbcs Input file name: remote-to-local single-byte conversion table. File extension is assumed to be SBC.

ldbcs Input file name: local-to-remote double-byte conversion table. File extension is assumed to be DBC.

rdbcs Input file name: remote-to-local double-byte conversion table. File extension is assumed to be DBC.

ltorsub
Four-character hexadecimal representation of double-byte substitution character for local-to-remote conversion. Any double-byte code points not represented in the table are transformed to this character.

rtolsub
Four-character hexadecimal representation of double-byte substitution character for remote-to-local conversion. Any double-byte code points not represented in the table are transformed to this character.

svx evx
In ASCII code pages, code points for single-byte characters and the first byte of double-byte characters are mutually exclusive. Within a

code page, ranges of byte values are reserved for the first byte of DBCS code points. ″svx″ and ″evx″ are two-character hexadecimal representations of the start value and end value for a range of code points. Up to three ranges can be specified on the command.

Specify only the first five parameters if creating conversion tables for a single-byte language.

An example of the command used for creating the English conversion table for MVS or VSE is shown below. Note that the MVS or VSE system is designated as the server (remote) system and the workstation is designated as the client (local) system, even when you are building a table for use on the MVS or VSE system.

```
HPTBLDCA ELACNENU AI E 437T037 037T437
```

An example of the command used for creating the Japanese conversion table for windows is the following:

```
HPTBLDCC ELACNJPN AI E 1041T290 290T1041 301T300 300T301 FEFE FCFC 81 9F E0 FC
```

Installing the Conversion Table on MVS or VSE

To install the MVS or VSE conversion table, upload the .370 file to the MVS or VSE system and assemble and link the file into a library accessible to host run-time services.

For MVS systems, the following parts in the Server for MVS, VSE, and VM sample library can be used to assemble and link the conversion table:

ELACVPLK
Sample procedure for assembling and link-editing a conversion table

ELACVJLK
Sample job that runs procedure ELACVPLK

For VSE systems, the following part in the Server for MVS, VSE, and VM library can be used to assemble and link the conversion table:

ELACVJLK
Sample job for assembling and linking the conversion table

Installing the Conversion Table on Windows

Once the C file is created, transfer the file to a Windows system and, using Borland’s C++ compiler, use the Borland project file to create a 32-bit DLL. Or, use your C++ compiler command for creating a 32-bit DLL.

The DLL contains a single function, CONVEP, which returns a pointer to the conversion table information defined in the program.

The conversion table is now ready for use by generated GUI client programs on Windows.

Bi-Directional Languages Attribute Conversion

Arabic and Hebrew are bi-directional (BIDI) languages in which national language characters are displayed right to left and Latin characters are displayed left to right. BIDI language attributes describe the way in which different types of characters (national language, Latin, numeric, and special characters) are stored and displayed in character string variables.

VisualAge Generator supports conversion of BIDI character strings from one attribute specification to another. This conversion is supported on Windows, OS/2 and AIX systems. BIDI attribute conversion can be performed in conjunction with or independently of code-page character conversion or numeric item data format conversion. You can convert the character strings in a record using an EZECONV special function call from the test facility and generated C++ programs), or you can request that character string parameters be converted on client/server calls to remote server programs from test facility, GUI clients, and generated C++ programs.

VisualAge Generator Developer does the following when requesting BIDI attribute conversion:

- Defines a conversion table that contains source and target BIDI attribute specifications in addition to code page and system type information.
- Specifies the conversion table name on the EZECONV call or on the CONTABLE option for the generation linkage table entry that defines how a server program is called.

Defining Conversion Tables for Bi-Directional Attributes

To build a conversion table with BIDI attribute specifications, use the HPTBLDBF.EXE utility. Use the following command to create a conversion table:

```
HPTBLDBF tname ct st csbcscp ssbcscp cbidi sbidi
```

where:

tname Output file name with a file extension is CTB.

ct Client (source) system type:

AI	ASCII Intel (Windows and OS/2)
AX	ASCII AIX or HP-UX
E	EBCDIC

st Server (target) system type:

AI	ASCII Intel (Windows and OS/2)
AX	ASCII AIX or HP-UX
E	EBCDIC

csbcscp
Client (source) system code page number (1- to 4-digit number)

ssbcscp
Server (target) system code page number (1- to 4-digit number)

cbidi Client (source) BIDI attributes (8-digit hexadecimal number)

A description of each digit in the BIDI attribute string (starting from the left) follows:

Digits	Description
1	Text manipulation. This attribute does not affect BIDI string conversion. Set the value to 8.
2	Text type **0** Visual **1** Implicit
3	Window orientation. This attribute does not affect BIDI string conversion. **0** Left to right **1** Right to left
4	Text orientation **0** Left to right **1** Right to left **2** Contextual
5	Numerals **0** Nominal **1** Pass through **2** National **3** Contextual
6	Symmetric swapping (ON = 1 OFF = 0) **0** Off **1** On
7-8	Character shape **00** Display shaped **01** Save shaped **10** Nominal **11** Initial **12** Middle **13** Final **14** Isolated

sbidi Server (target) BIDI attributes (8-digit hexadecimal number)

See the previous option, *cbidi*, for a description of each digit in the BIDI attribute string.

The system types control the conversion of the numeric item format. The code page numbers control code page translation for character items when a conversion is performed. The BIDI attributes control BIDI attribute conversion for character items.

The following command builds a conversion table that supports Arabic ASCII-to-EBCDIC conversion with BIDI attribute conversion for an OS/2 client calling an MVS server:

```
HPTBLDBF CNAE1ARA AI E 864 420 81110000 80110000
```

The following command builds a conversion table that supports BIDI attribute conversion on OS/2. Code page and numeric items conversions are not performed because the same system type and code page is specified for both source and target systems.

```
 HPTBLDBF CNAA1ARA AI AI 864 864 81110000 80112000
```

BIDI Attribute Conversion Utility for ESF Files Downloaded From the Host

BIDI attribute conversion cannot be performed on an ESF file by a file transfer program because the file transfer program cannot separate the BIDI constant strings in the ESF file from the rest of the ESF information. Use the following process to perform BIDI attribute conversion on an ESF file that is being moved from a CSP 4.1 system to VisualAge Generator Developer:

1. Download the ESF file. Use PCOMM or the file transfer program to perform EBCDIC to ASCII code page translation, but do not perform BIDI attribute conversion in this step. Table 59 lists the source and target code pages based on language and system.

Table 59. Source and Target Code Pages for EBCDIC to ASCII translation

Language	VisualAge Generator Developer System	Host Code Page	ASCII Code Page
Arabic	Windows NT	420	864
Arabic	OS/2	420	864
Hebrew	Windows NT	424	1255
Hebrew	OS/2	424	862

2. Define a BIDI conversion table specifying the host BIDI attributes as the server attributes and the developer system BIDIB attributes as the client attributes. Specify source and target code pages for the conversion table based on language and system as shown in Table 60 on page 448.

Table 60. Client/Server Source and Target Code Pages

Language	VisualAge Generator Developer System	Client Code Page	Server Code Page
Arabic	Windows NT	1256	864
Arabic	OS/2	864	864
Hebrew	Windows NT	1255	1255
Hebrew	OS/2	862	862

3. Use the HPTCNESF utility to perform BIDI attribute conversion by entering:

   ```
   HPTCNESF  input-esf-file-name output-esf-file-name conversion-table-name
   ```

   For example, if the downloaded ESF file is csp1.esf and the conversion table is CNHEBESF, enter:

   ```
   HPTCNESF CSP1.ESF  CSP1OUT.ESF CNHEBESF
   ```
4. Import the output ESF file (CSP1OUT.ESF in the previous example) into a VisualAge Generator application or package.

Using Conversion Tables with EZECONV

The following example illustrates how to code the EZECONV special function in a program:

```
MOVE "CNAA1ARA" TO CONVTAB;
CALL EZECONV MYRECORD,'L',CONVTAB;
```

EZECONV converts the data items in record MYRECORD from local (client) format to remote (server) format as specified in conversion table CNAA1ARA.CTB. To convert in the opposite direction, specify 'R' instead of 'L' on the EZECONV call.

Using Conversion Tables on Calls to Server Programs

The following example illustrates a call to program MYSERVER as coded in a client program. Two parameter records are passed on the call.

```
CALL MYSERVER MYRECORD1, MYRECORD2
```

A linkage table entry for MYSERVER indicates to test, generation, and VisualAge Generator Server that MYSERVER is a remote server program. The CONTABLE option on the entry specifies the conversion table to use to convert the data in the parameter records on the call. The data is converted from local (client) to remote (server) format before the server is called, and again from remote to local format after the server returns.

The following linkage table entry example specifies that MYSERVER is a remote program called using the CICS client ECI middleware. Parameter conversion is done as specified in conversion table CNAE1ARA.

```
:calllink applname=MYSERVER linktype=REMOTE remotecomtype=CICSCLIENT
contable=CNAE1ARA.
```

Appendix D. DBCS and Client/Server Processing

Considerations for programs containing DBCS data are described in this section.

Check SO/SI Map Edit

MIX (mixed single and double byte) data values require fewer bytes for storage on all workstation systems because the ASCII DBCS format does not use shift-in/shift-out escape characters for delimiting DBCS data.

If your client/server program design permits mixed data entry on a workstation that it then stores in a file or database on the mainframe, use the Check SO/SI Space map item edit to ensure that the mixed values entered on a workstation can be converted to the EBCDIC SO/SI format for mixed data and still fit in a field of the same length. Values that do not fit are truncated at a valid DBCS character boundary on conversion.

For more DBCS considerations, refer to *VisualAge Generator Design Guide*

Converting Variable Length Records with MIX Items

SO/SI escape characters are deleted from MIX items on EBCDIC-to-ASCII format conversion and inserted in MIX items on ASCII-to-EBCDIC conversion. If a variable length record that contains MIX items is being converted, and the current record end as indicated by the record length lies within a MIX item, the record length is adjusted on conversion to reflect the insertion or deletion of SO/SI characters.

Appendix E. VisualAge PowerServer APIs

This section contains information that applies to using VisualAge PowerServer application programming interfaces to call VisualAge Generator server programs from client programs developed using other tools. You should be familiar with the following chapters before reading the API information:

- Chapter 2. Introduction to Client/Server Processing with Synchronous Calls
- Appendix B. Linkage tables
- Appendix C. Converting Between Client and Server Data Formats

For more information on calling servers using specific communications protocols, refer to the chapter on configuring your platform.

The VisualAge PowerServer APIs provide a common C interface for performing the following functions:

- Initializing a communications session
- Calling a server program on a remote system
- Committing or rolling back a unit of work that is controlled by the client system
- Ending the communications session
- Retrieving an error message when a function indicates it has not completed successfully

Client Systems

Client programs can call the APIs on any of the following systems:

- OS/2 Warp Version 4.0 or later
- AIX Version 4.1.3 or later
- Windows 95 (WIN32 interface only)
- Windows NT 4.0 or later

Note: VisualAge Generator Developer, VisualAge Generator Server, or VisualAge Generator Common Services must be installed on the client system.

Server Systems

Client programs can use the APIs to call servers running in any of the following environments:

- CICS for MVS/ESA
- AIX

- CICS for AIX
- CICS for OS/2
- CICS for Solaris
- CICS for VSE/ESA
- CICS for Windows NT
- HP-UX
- IMS
- OS/2
- OS/400
- Solaris
- VM/ESA
- Windows NT

Note: The installation of VisualAge Generator Server or VisualAge Generator Server for MVS, VSE, and VM might be required on the server system.

Supported Middleware

Client programs can call server programs via the following middleware:
- CICS Client for CICS server programs
- Client Access/400 for OS/400 server programs
- APPC/MVS for IMS server programs
- DCE RPC for C++ programs running on OS/2, AIX and Windows NT.
- TCP/IP for C++ programs running on OS/2, AIX and Windows NT, HP-UX, Solaris and VM/ESA.
- LU2 for COBOL programs running on CICS for MVS/ESA.

Note: The middleware options available vary by client and server platform combinations. See Table 2 on page 12 for additional information.

Coding the Client Program

File **cso2api.h** in the **\include** subdirectory for VisualAge Generator Developer and VisualAge Generator Server contains the data item, data structure, and function prototype definitions for the VisualAge PowerServer APIs.

A client program must establish a communications session using the CMINIT function before calling any other functions.

The data declarations for the data passed to the server program on the CMCALL function must be compatible with the parameters defined to the server program, The length of each parameter must be the same for both the client and the server programs.

Data Type Descriptions

VisualAge PowerServer calls use both elementary and structure data types.

The elementary data types are:

CMCHAR
: Single byte character

CMCHAR4
: String of four characters

CMEMSG
: 70-character array containing null terminated string

CMNAME
: Character array containing null terminated string

CMMSG
: Null terminated string

CMPARM
: Argument passed to server program

CMSHORT
: Short integer

CMLONG
: Long integer

CMCC
: Long integer

The structure data types are:

CMCOD
: Call options descriptor

CMCVOD
: Conversion options descriptor

CMCOMP
: Completion status descriptor

CMDESC
: Parameter structure descriptor

Structure Definitions

This section describes the structure definitions.

CMCOD - Call Options Descriptor

The call options descriptor specifies options that control how a remote program is called.

Fields in the descriptor structure are:

CMCHAR4
: ***StrucId***

Structure identifier.

The value must be:

CMCOD_STRUC_ID = ″COD″

CMLONG
Version

Structure version number

The value must be:

CMCVOD_VERSION_1 = 1

CMLONG
Protocol

The communications protocol used to communicate with the client program. Valid values are:

CMST_RUNTIME_BIND = 0
The communications protocol is read from the linkage table at runtime. In addition, the following option values are read from the linkage table and any corresponding option specified on the CMCALL is ignored:
LuwType
AppType
Parmform
ConversionTable
Location
Serverid
Library

CMST_ECI_C2 = 7
CICS OS/2 External Call Interface

CMST_ECI_CM = 8
CICS Client External Call Interface

CMST_CA400 = 9
Client Access/400

CMST_APPC_IMS = 11
LU 6.2 connection to IMS message processing region

CMST_DCE = 13
Distributed Computing Environment Remote Procedure Call (DCE RPC), no authorization checking

CMST_DCE_SECURE = 14
Distributed Computing Environment Remote Procedure Call (DCE RPC) with authorization checking

CMST_PACBASE = 15
PACKBASE

CMST_TCPIP = 19
TCP/IP

CMST_LU2 = 20
LU2

CMLONG
LuwType

Logical unit of work type. Values are:

CMLUW_CLIENT = 0
Unit of work is under client control.

Server updates are not committed or rolled back until the client requests commit or rollback using the CMCOMMIT or CMROLLBK services. Server programs cannot request commit or rollback.

Environments which do not support client controlled unit of work will ignore this value. See Table 2 on page 12 for the environments which support client controlled unit of work. For those environments which do not support client controlled unit of work, server unit of work will be used.

CMLUW_SERVER = 1
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 request rollback.

CMLONG
AppType

Remote program type

CMLUW_VG = 0
The called program is a generated VisualAge Generator program which was generated with a linktype of remote. An additional parameter is automatically passed to the server to allow the server to return an error code to the middleware if the server program ends abnormally.

CMLUW_NONVG = 1
The called program was developed using a tool other than VisualAge Generator. Only the parameters passed on the CMCALL service are passed to the called program.

CMLONG
Parmform

Parameter format

This option is supported only when calling via CICS Client ECI. It is ignored for all other types of middleware.

CMPF_COMMPTR = 0
The server program expects to be called using the CSP/AE parameter passing convention that uses pointers in the COMMAREA. Use only with MVS CICS, VSE CICS, or CICS OS/2 server programs that were generated or coded to use this parameter passing convention.

CMPF_COMMDATA = 1
The server program expects to receive the parameter values in the CICS COMMAREA. The parameter values passed to CMCALL are moved into a single buffer, each value adjoining the previous value without regard for boundary alignment. On return from the remote call, the values returned in the output buffer are moved back to the corresponding parameters passed to the CMCALL.

CMNAME
ConversionTable[9]

Conversion table name

Specifies the name of the conversion table used to perform automatic data conversion on the call to the remote program. The name is a 9-byte character array containing a null-terminated character string.

Some names have a special meaning:

*** (asterisk)**
Conversion is performed on the client using the default conversion table. You must enclose the asterisk in single quotes.

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 ELACNxxx, where xxx is the national language code specified in environment variable EZERNLS. If EZERNLS is not specified, the default national language code is ENU.

Note: The default conversion tables assume that the server platform is MVS, VSE, OS/400, or VM.

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, HP-UX, and Solaris servers; and vice versa, when both the client and the server are running under the same code page.

NONE
No conversion is performed.

Refer to Appendix C. Converting Between Client and Server Data Formats in the *VisualAge Generator Client/Server Communications Guide* manual for more information on conversion and conversion tables.

CMNAME
Location[20]

Protocol dependent server system name. The name is a 20-byte character array containing a null-terminated character string.

The following table shows the meaning of the identifier by protocol and the default value if a name is not specified (null string).

Protocol	Meaning of location	Default value
TCPIP	TCP/IP hostname of server	No default
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
APPCIMS	CPIC side information identifier. The side information specifies: • Partner LU Alias • Transaction Program Name • Mode Name	No default
CA/400	AS/400 system identifier	The managing OS/400 system

The value is ignored for other protocols.

CMNAME
ServerId[20]

Protocol dependent server channel or transaction name. The name is a 20-byte character array containing a null-terminated character string.

The following table shows the meaning of the identifier by protocol and the default value if a name is not specified (null string).

Protocol	**Meaning of Server Identifier**
TCP/IP	Name of TCP/IP service port as defined in the etc\services file.
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.
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.

The value is ignored for other protocols.

CMNAME
LinkageTableName[80]

Linkage table file name identifying the linkage table to be used if runtime bind is specified for the Protocol parameter.

If not specified (null string) or not found, the linkage table file name is obtained from environment variable CSOLINKTBL.

If the name is not fully qualified, the CMCALL service uses the current DPATH search path to find the file.

CMNAME
Library[20]

OS/400 program library name. The name is a 20-byte character array containing a null-terminated character string.

This value is used only with the Client Access/400 protocol. It specifies the name of the OS/400 library that contains the called program. The default value is the program name if the array contains a null string.

CMNAME
ExtName[40]

The name of the program that is called. The EXTERNALNAME value can be more than 8 characters long.

This value is used only with the Client Access/400 protocol. The default value is the APPLNAME.

CMCOMP - Completion Status Descriptor

The completion status descriptor returns status information that indicates whether a service call completed successfully or not, and describes the reasons for unsuccessful completions.

Fields in the descriptor structure are:

CMCC
CompCode

Completion code. Possible values are:

CMCC_OK = 0
Successful completion.

CMCC_WARN = 4
Warning (partial completion).

Reason code and message inserts give additional information.

CMCC_ERROR = 8
Call failed.

Reason code and message inserts give additional information.

CMCC_12BYTE_ERROR = 12
Host or work group services detected a terminating error in the called program.

CMCC_SECURITY = 16
The user is not authorized to call the remote program.

Reason code and message inserts give additional information.

CMLONG
Reason

The reason code gives further information on reasons for failing completion codes. Each four digit reason code is associated with error message number CSOxxxxE (xxxx is the reason code). Refer to the description in the *VisualAge Generator Messages and Problem Determination Guide* manual. The message description describes the error associated with a reason code.

CMLONG
EmsgCount

Error insert count. The number of message inserts that follow in the error insert array.

CMEMSG
Emsg[10]

Error insert array. Each array element contains a message insert for the error message associated with the reason code. Each message insert is a null terminated string with a max length of 72 characters (including the null terminator).

Note: The reason code, insert count, and insert array are set only if the completion code is not OK.

CMDESC - Parameter structure description

The parameter descriptor is a variable length structure that defines the format of the parameter data. The descriptor is used in moving the parameter data to the transmission buffer and in converting the data format of the parameter from the client format to server format or vice versa.

The description is sequence of substructures. There are four types of substructures:

CMPARMLEN
Parameter Length Descriptor

CMITEM
Parameter Item Descriptor

CMOCCURS
Occurring Structure Descriptor

CMPARMEND
Parameter Descriptor End Marker

CMPARMLEN - Parameter Length Descriptor

The length of the parameter value. This is the length of data moved to the transmission buffer. CMPARMLEN must be the first substructure in the parameter descriptor.

Fields in the descriptor structure are:

CMCHAR
ParmLengthIndicator

CONV_MAX_LENGTH = 0xF3

CMLONG
ParmLength

Length of parameter in bytes

CMITEM - Parameter Item Descriptor

An item descriptor defines the type and length of a field within the parameter. There must be one item descriptor for each low level item in the parameter structure definition

Fields in the descriptor structure are:

CMCHAR

ParmItemIndicator

The item indicator indicates item type. The type indicates what format conversion should be used on the item. Item types correspond to VisualAge Generator item definition types and can have the following values:

DATA_BIN = 0x01

DATA_CHA = 0x02

DATA_DBCS = 0x03

DATA_HEX = 0x04

DATA_MIX = 0x05

DATA_NUM = 0x06

DATA_NUMC = 0x07

DATA_PACK = 0x08

DATA_PACKF = 0x09

DATA_PSB = 0x0A

Contiguous files of type DATA_CHA can be described with a single item descriptor.

Contiguous fields of these types can be defined with a single item descriptor, except for DATA_PSB. DATA_PSB requires an item descriptor defined as a 12 byte array of nulls.

All other item types require an item descriptor for each elementary item in the parameter structure.

No conversion is performed for the following item types:
- DATA_NO_CONV
- DATA_HEX
- DATA_PACK
- DATA_PACKF

CMSHORT

ItemLength

Length of item in bytes

CMOCCURS - Occurring Structure Descriptor

If the parameter structure includes a multiply occurring structure (an array) then this structure must immediately precede the sequence of item descriptors that represent the elementary fields that make up the array. The descriptor identifies the number of occurrences plus the number of item descriptors represented within each occurrence.

Fields in the descriptor structure are:

CMCHAR
: *OccurrencesIndicator*

  OCC_STR = 0xFD

CMSHORT
: *OccurringItems*

  Number of item descriptors in the occurring structure

CMSHORT
: *Dimension*

  Number of occurrences in the occurring structure

CMPARMEND - Parameter Descriptor End Marker

A marker for the end of the parameter description.

Fields in the descriptor structure are:

CMCHAR
: *ParmEndIndicator*

  END_DESC = 0xFF

Example - Parameter and Descriptor Definitions in C

The following example shows examples of how to code parameters and parameter descriptors for a simple, single item parameter, a simple structure, and a complex structure which includes a multiply occurring array.

The structures in the example must be compiled to insure that the items in the structure are aligned on byte boundaries. You can compile the example with compiler option **/Sp1** for IBM VisualAge C++.

```
#include "cso2api.h"


typedef struct
 {
 CMCHAR parmLengthIndicator;
 CMLONG parmLength;
 }
 CMPARMLEN ;

typedef struct
 {
```

```
 CMCHAR  itemIndicator;
 CMSHORT itemLength;
 }
 CMITEM ;

typedef struct
 {
 CMCHAR  occursIndicator;
 CMSHORT occursItemCount;
 CMSHORT occursDimension;
 }
 CMOCCURS ;

typedef struct
 {
 CMCHAR parmEndIndicator;
 }
 CMPARMEND ;


/* Example 1 - Simple parameter and descriptor definition */

char key[6]; /* Parameter definition */

struct {
 CMPARMLEN parmlen ;
 CMITEM keyDesc ;
 CMPARMEND parmend ;
 } keyDescriptor = {
 {CONV_MAX_LENGTH, sizeof(key)},
 {DATA_CHA,sizeof(key)},
 END_DESC
 } ;  /* Parameter descriptor */

/* Example 2 - Simple structure parameter and descriptor definition */

typedef struct {
 char name[10];
 long number;
 } PART ;

PART part ; /* Parameter definition */

struct {
 CMPARMLEN parmlen ;
 CMITEM nameDesc ;
 CMITEM numberDesc ;
 CMPARMEND parmend ;
 } partDescriptor = {
 {CONV_MAX_LENGTH, sizeof(part)},
 {DATA_CHA,sizeof(part.name)},
 {DATA_BIN,sizeof(part.number)},
 END_DESC
 } ;  /* Parameter descriptor */
```

```
/* Example 3 - Complex structure parameter and descriptor definition */

typedef struct {
 char cName[10];
 long cNumber;
 PART cPart[5];
 } COMPONENT ;

COMPONENT component ; /* Parameter definition */

struct {
 CMPARMLEN parmlen ;
 CMITEM cNameDesc ;
 CMITEM cNumberDesc ;
 CMOCCURS cPartOccurs;
 CMITEM nameDesc ;
 CMITEM numberDesc ;
 CMPARMEND parmend ;
 } componentDescriptor = {
 {CONV_MAX_LENGTH, sizeof(component)},
 {DATA_CHA,sizeof(component.cName)},
 {DATA_BIN,sizeof(component.cNumber)},
 {OCC_STR,2,5},
 {DATA_CHA,sizeof(component.cPart[0].name)},
 {DATA_BIN,sizeof(component.cPart[0].number)},
 END_DESC
 } ;  /* Parameter descriptor */

 /* Parameter Pointer Array */

 CMPARM *paData[3] =
      {
      (char*)key,
      (char*)&part,
      (char*)&component
      } ;

 /* Descriptor Pointer Array */

 CMDESC *paDesc[3] =
      {
      (char*)&keyDescriptor,
      (char*)&partDescriptor,
      (char*)&componentDescriptor
      } ;
```

Call Descriptions

This section describes VisualAge PowerServer API functions:

CMCALL
Call Remote Server Program

CMCLOSE
: Close Communications Session

CMCOMMIT
: Commit Unit of Work

CMGETERROR
: Get Error String

CMINIT
: Initialize Communications Session

CMROLLBK
: Rollback Unit of Work

CMCALL — Call Remote Server Program

The CMCALL call makes a call to a remote server program.

```
CMCC CMCALL ( CMLONG Hconn,
              CMNAME *Applname,
              CMLONG ParmCount,
              CMPARM **Parameter,
              CMDESC **Description,
              CMCOD *Cmcod,
              CMCOMP *Cmcomp)
```

Parameters

Hconn (CMLONG) — input
: This connection handle represents the connection to the VisualAge PowerServer communications session. The handle was returned by a previous CMINIT call.

Applname (CMNAME *) — input
: This server program name parameter specifies the name of the server program that is being called. This name is used to determine which linkage table CALLLINK entry will be used for routing the remote call.

  The name is a null-terminated character string with a maximum length of eight characters plus the null terminator.

ParmCount (CMLONG) — input
: This parameter specifies the number of parameters that are being passed on the remote call to the server program.

Parameter (CMPARM **) — input
: This parameter is an array of pointers, one pointer to each parameter to be passed to the server program.

Description (CMDESC **) — input
: This parameter is an array of pointers, one pointer for each parameter to be passed to the server program. The pointer points to the

parameter descriptor that describes the format of the parameter data. The description is used in moving the parameter data to the communications buffer, and in converting the data between client and server system format. See "CMDESC - Parameter structure description" on page 462 for a description of the **Description** parameter.

Cmcod (CMCOD *) — input
This call descriptor structure is used to describe the options for the remote call. All of the attributes that can be specified in the linkage table can be specified in this structure instead. See "CMCOD - Call Options Descriptor" on page 455 for a description of the CMCOD structure.

Cmcomp (CMCOMP *) — output
This completion code structure is used to pass back a completion code, error reason code, and message insert information. See "CMCOMP - Completion Status Descriptor" on page 461 for a description of the CMCOMP structure.

Usage

- There is a limit of 32567 bytes of parameter data that can be passed on any CMCALL.
- If a linkage table name is specified in the CMCOD structure and the all of the call descriptor options are not specified in the CMCOD structure, the VisualAge PowerServer code will search the DPATH (for OS/2 and AIX) or PATH (for Windows) for the linkage table specified in the CMCOD structure. If no linkage table is specified in the CMCOD structure or it can not be found in the DPATH, then the CSOLINKTBL environment variable will be used to locate the linkage table to be used.

CMCLOSE — Close Communications Session

The CMCLOSE call terminates a VisualAge PowerServer communications session.

```
CMCC CMCLOSE ( CMLONG Hconn,
               CMCOMP *Cmcomp)
```

Parameters

Hconn (CMLONG) — input
This connection handle represents the connection to the VisualAge PowerServer communications session. The handle was returned by a previous CMINIT call.

Cmcomp (CMCOMP *) — output
This completion code structure is used to pass back a completion code, error reason code, and message insert information. See "CMCOMP - Completion Status Descriptor" on page 461 for a description of the CMCOMP structure.

Usage

- A usage count is kept for each of the communications sessions. The usage count decrements on each CMCLOSE call. When the usage count reaches 0, the communications session is closed and the communications session handle is not longer valid.

CMCOMMIT — Commit Unit of Work

The CMCOMMIT call propagates a commit to the server platforms, as appropriate.

```
CMCC CMCOMMIT ( CMLONG Hconn,
                CMCOMP *Cmcomp)
```

Parameters

Hconn (CMLONG) — input
This connection handle represents the connection to the VisualAge PowerServer communications session. The handle was returned by a previous CMINIT call.

Cmcomp (CMCOMP *) — output
Completion code structure.

This structure is used to pass back a completion code, error reason code, and message insert information. See "CMCOMP - Completion Status Descriptor" on page 461 for a description of the CMCOMP structure.

Usage

CMCOMMIT will only have an affect when an CMCALL has been made via the CICS Client, CICS for OS/2 ECI, or Client Access/400 middleware products and client unit of work was specified. In all other cases, an CMCOMMIT call results in an immediate return with no action taken.

CMGETERROR — Get Error String

The CMGETERROR call returns the VisualAge PowerServer communications error string.

```
CMCC CMGETERROR ( CMCOMP *Cmcomp,
                  CMLONG *CmErrorStringLen,
                  CMMSG *CmErrorString)
```

Parameters

Cmcomp (CMCOMP *) — input
This completion code structure was returned from a prior VisualAge PowerServer call. It contains the error reason code and message insert information that will be used to build the error string returned from the CMGETERROR call.

CmErrorStringLen (CMLONG *) — input/output
Maximum length of error string.

CmErrorString (CMMSG *) — output
: VisualAge PowerServer error string. CmErrorString is a pointer to a null terminated string.

Usage

When the maximum length of an error string (CmErrorStringLen) is less than the length of the returned formatted error string, the following occurs:

1. CMErrorString buffer holds a truncated formatted error string.
2. CmErrorStringLen is updated to a length value that can hold the entire formatted error string.
3. A CMCC value of 4 is returned.

You can either use the truncated formatted error string returned from the API call, thereby ignoring the CMCC value of 4; or you can allocate a larger CmErrorString buffer based on the updated value of CmErrorStringLen and call the API again.

CMINIT — Initialize Communications Session

The CMINIT call starts a VisualAge PowerServer communications session. CMINIT provides a communications session handle that is used by the program on subsequent VisualAge PowerServer communications calls.

```
CMCC CMINIT ( CMLONG *Hconn,
              CMCOMP *Cmcomp,
              void *winHandle,
              CMLONG *forceNewHandle)
```

Parameters

Hconn (CMLONG *) — output
: This connection handle represents the connection to the VisualAge PowerServer communications session. It must be specified on all subsequent VisualAge PowerServer calls issued by the program. It ceases to be valid when the CMCLOSE call is issued, or when the process that initialized the handle terminates.

Cmcomp (CMCOMP *) — output
: This completion code structure is used to pass back a completion code, error reason code, and message insert information. See “CMCOMP - Completion Status Descriptor” on page 461 for a description of the CMCOMP structure.

winHandle (void *) — input
: This window handle parameter is not used and should be set to NULL.

forceNewHandle (CMLONG *) — input
: This option controls the action of CMINIT. The **forceNewHandle** parameter controls the assignment of the connection handle. Valid values for this parameter are:

TRUE Always return a handle to a new communications session.

FALSE
return a handle to an existing communications session if one already exists in the current process.

```
existing session | forceNewHandle |  CMINIT action
-----------------+----------------+------------------------------
       No        |      ---       |  return handle to new session
                 |                |
       Yes       |      TRUE      |  return handle to new session
                 |                |
       Yes       |      FALSE     |  return handle to existing
                 |                |  session
```

Usage

- If an CMINIT call is made with **forceNewHandle** set to FALSE and there are two or more existing communications sessions in the current process, there is no guarantee as to which of the existing communications sessions will be returned from the call.
- A usage count is kept for each of the communications sessions. The usage count increments by one each time that the session is returned on an CMINIT call. There should be a corresponding CMCLOSE call for each CMINIT call made.
- Memory allocated during the CMINIT call is freed when the communications session is closed. Failure to call CMCLOSE will result in memory leaks in your program.

CMROLLBK — Rollback Unit of Work

The CMROLLBK call propagates a rollback to the server platforms, as appropriate.

```
CMCC CMROLLBK ( CMLONG Hconn,
                CMCOMP *Cmcomp)
```

Parameters

Hconn (CMLONG) — input
This connection handle represents the connection to the VisualAge PowerServer communications session. The handle was returned by a previous CMINIT call.

Cmcomp (CMCOMP *) — output
This completion code structure is used to pass back a completion code, error reason code, and message insert information. See "CMCOMP - Completion Status Descriptor" on page 461 for a description of the CMCOMP structure.

Usage

CMROLLBK will only have an affect when an CMCALL has been made via the CICS Client, CICS for OS/2 ECI, or Client Access/400 middleware products and client unit of work was specified. In all other cases, an CMROLLBK call results in an immediate return with no action taken.

Compiling and Linking the Client Program

This section describes compiling and linking client programs.

OS/2

You must use a 32-bit compiler. These compile and link options have been tested using the IBM VisualAge C++ V3.0 compiler.

Compile options:

```
/D__OS2__
```

Link libraries:

```
cso40api.lib
```

AIX

These compile and link options have been tested using the IBM C Set ++ for AIX V3.1 compiler.

Compile options:

```
/D_AIX
```

Link libraries:

```
libcso40api.a
```

Windows NT and Windows 95

These compile and link options have been tested using Microsoft Visual C++ V4.0 compiler.

Compile options:

```
/D_Windows
/D_WINDOWS
/DCSOWIN32
```

Link libraries:

```
cso40api.lib
```

User Authentication

VisualAge PowerServer APIs use the same user authentication procedures as VisualAge Generator client/server programs. See “User Authentication” on page 21 for more information.

Error Handling

VisualAge PowerServer APIs return an error description in structure CMCOMP if a function call does not complete successfully. Function CMGETERROR can be called to turn the error description into an error message.

Additional trace information for VisualAge PowerServer API calls by setting the following environment variables:

CSODIR
Specifies the directory where the CSO.INI file is located. By default, this location is the root directory where VisualAge Generator Common Services is installed for OS/2 or Windows environments, or the root directory where VisualAge Generator Server is installed for AIX, HP-UX, and Solaris environments.

CSO_DUMP_CONV
The CSO_DUMP_CONV environment variable is available for debugging VisualAge Generator programs and should only be set upon instruction from VisualAge Generator Support. When CSO_DUMP_CONV=1, trace entries are produced that document the parameters being passed and their values prior to and after the middleware support converts the data. The conversion table used is also documented in the trace. This information is written to the file CSODUMP.OUT.

CSO_DUMP_DATA
The CSO_DUMP_DATA environment variable is available for debugging VisualAge Generator programs and should only be set upon instruction from VisualAge Generator Support. When CSO_DUMP_DATA=1, trace entries are produced that document the linkage table parameters used for the server call, the parameters being passed, and the values of these parameters. The descriptor and value information is documented prior to and after the call to the server. This information is written to the file CSODUMP.OUT.

CSOTIMEOUT
Specifies the length of time in seconds after which a time-out error occurs if the server does not respond to the client. The default value is 30.

CSOTROPT
Specifies the level of trace information collected.

CSOTROPT=1
Only errors are collected in the trace file. This is the default if CSOTROPT is not set.

CSOTROPT=2
All traces are collected in the trace file

CSOTROUT
Specifies the trace file name. The default name is csotrace.out in your current directory.

Appendix F. Run-time configurations for VisualAge Generator

The following tables describe run-time configurations supported by VisualAge Generator:

- VisualAge Generator GUI/TUI to Server
- Java Applet/Servlet to Server
- Enterprise Java Beans
- Websphere rapid application development (UI Record)
- Usage and Platform Matrix
- From/To Protocol Matrix

You can use the tables in this section to determine available platforms and protocol for a VisualAge Generator two or three tier system configuration based on the client usage you select. For example, suppose you want to configure the following system:

- A Java GUI Application client running on the Windows 98 platform
- A second tier server running on the CICS for Windows NT platform
- A third tier server running on the CICS for MVS/ESA platform

You can:

1. In Table 61 on page 476, find Java GUI Application in the client **Usage** column.
2. In Table 65 on page 483, in the **Java GUI Application** column, confirm that Windows 98 is a supported platform for a Java GUI Application.
3. In Table 66 on page 484, in the **From** column, find the client platform, Windows 98.
4. In Table 66 on page 484, in the **To** column, find the 2nd tier server, CICS for Windows NT.
5. In Table 66 on page 484, confirm that there is a supported protocol from Windows 98 to CICS for Windows NT. CICS CLIENT is the supported protocol from Windows 98 to CICS for Windows NT.
6. In Table 66 on page 484, in the **From** column, find the 2nd tier server, CICS for Windows NT.
7. In Table 66 on page 484, in the **To** column, find the 3rd tier server, CICS for MVS/ESA.
8. In Table 66 on page 484, confirm that there is a supported protocol from CICS for Windows NT to CICS for MVS/ESA. CICS DPL is the supported protocol from Windows 98 to CICS for Windows NT.

Table 61. VisualAge Generator GUI/TUI to Server

Client		**Communications**	**Second Tier Server**		**Communications**	**Third Tier Server**	
Usage	**Required Software**	See Table 65 on page 483 to determine the platforms that are supported for your client usage. See Table 66 on page 484 for the protocol options available from your client to your 2nd tier server.	**Platform**	**Required Software**	See Table 66 on page 484 for the protocol options available from your 2nd tier server to your 3rd tier server.	**Platform**	**Required Software**
ST GUI	CSO		Windows NT, Windows 2000	WGS		Windows NT, Windows 2000	WGS
Java GUI Application	CSO^{1}or WGS^1		OS/2	WGS		OS/2	WGS
C++ TUI	WGS		AIX	WGS		AIX	WGS
COBOL TUI	WGS^2 or HS		Solaris	WGS		Solaris	WGS
ITF	Developer		HP/UX	WGS		HP/UX	WGS
			VM/ESA	HS		VM/ESA	HS
			IMS	HS		IMS	HS
			AS/400	HS400		AS/400	HS400
			CICS for Windows NT	WGS		CICS for Windows NT	WGS
			CICS for OS/2	WGS		CICS for OS/2	WGS
			CICS for AIX	WGS		CICS for AIX	WGS
			CICS for Solaris	WGS		CICS for Solaris	WGS
			CICS for MVS/ESA	HS		CICS for MVS/ESA	HS
			CICS for VSE/ESA	HS		CICS for VSE/ESA	HS

Table 61. VisualAge Generator GUI/TUI to Server (continued)

Client	Communications	Second Tier Server	Communications	Third Tier Server
Notes: CSO = VisualAge Generator Common Services (CSO code ships with VisualAge Generator Developer and can be redistributed on the Windows and OS/2 platforms) Developer = VisualAge Generator Developer HS = VisualAge Generator Server for MVS, VSE, and VM HS400 = VisualAge Generator Host Server for AS/400 WGS = VisualAge Generator Server for OS/2, Windows NT, AIX, HP-UX, and Solaris [1] CSO for Windows and OS/2, WGS for AIX [2] WGS for CICS for OS/2 Programs generated for the following platforms are not available as clients or 2nd tier of a 3 tier configuration: AS/400 HP/UX IMS VM/ESA				

Table 62. Java Applet/Servlet to Server

Client	Communications	Web Server		Communications	VisualAge Generator Server	
Web Browser downloading a Java applet on any platform	RMI	**Platform**	**Required Software**	See Table 66 on page 484 for the protocol options available from your 2nd tier server to your 3rd tier server.	**Platform**	**Required Software**
Web Browser accessing a Java servlet URL on any platform	HTTP	Windows NT, Windows 2000	CSO		Windows NT, Windows 2000	WGS
		OS/2[1]	CSO		OS/2	WGS
		AIX	WGS		AIX	WGS
		Solaris	WGS		Solaris	WGS
		AS/400	HS400		HP/UX	WGS
		OS/390 Unix System	HS		VM/ESA	HS
					IMS	HS
					AS/400	HS400
					CICS for Windows NT	WGS
					CICS for OS/2	WGS
					CICS for AIX	WGS
					CICS for Solaris	WGS
					CICS for MVS/ESA	HS
					CICS for VSE/ESA	HS

Table 62. Java Applet/Servlet to Server (continued)

Client	Communications	Web Server	Communications	VisualAge Generator Server
Notes: CSO = VisualAge Generator Common Services (CSO code ships with VisualAge Generator Developer and can be redistributed on the Windows and OS/2 platforms) HS = VisualAge Generator Server for MVS, VSE, and VM HS400 = VisualAge Generator Host Server for AS/400 WGS = VisualAge Generator Server for OS/2, Windows NT, AIX, HP-UX, and Solaris [1] Not valid for Java servlet				

For the following scenario, the ″Client″ is the Web Server where the EJB call is being made, not the browser running the Applet.

Table 63. Enterprise Java Beans

Web Server		**Communications**	**Enterprise Java Server**		**Communications**	**VisualAge Generator Server**	
Usage	**Required Software**	Enterprise Java Server (EJS) Communications **Note:** Actual protocol used varies depending on the EJS used. Websphere is currently using IIOP.	**Platform**	**Required Software**	See Table 66 on page 484 for the protocol options available from your 2nd tier server to your 3rd tier server.	**Platform**	**Required Software**
Java servlet accessed from a browser	WGS*, HS*, CSO*, or HS400		Windows NT, Windows 2000	CSO		Windows NT, Windows 2000	WGS
			OS/2	CSO		OS/2	WGS
			AIX	WGS		AIX	WGS
			Solaris	WGS		Solaris	WGS
			AS/400	HS400		HP/UX	WGS
			OS/390 Unix System	HS		VM/ESA	HS
						IMS	HS
						AS/400	HS400
						CICS for Windows NT	WGS
						CICS for OS/2	WGS
						CICS for AIX	WGS
						CICS for Solaris	WGS
						CICS for MVS/ESA	HS
						CICS for VSE/ESA	HS

Table 63. Enterprise Java Beans (continued)

Web Server	Communications	Enterprise Java Server	Communications	VisualAge Generator Server

Notes:
CSO = VisualAge Generator Common Services (CSO code ships with VisualAge Generator Developer and can be redistributed on the Windows and OS/2 platforms)
HS = VisualAge Generator Server for MVS, VSE, and VM
HS400 = VisualAge Generator Host Server for AS/400
WGS = VisualAge Generator Server for OS/2, Windows NT, AIX, HP-UX, and Solaris
 	
* WGS for AIX, HS for OS/390 Unix System, and CSO for Windows and OS/2.				

Table 64. Websphere rapid application development (UI Record)

Client	Communications	Web Server		Communications	VisualAge Generator Server	
		Platform	**Required Software**		**Platform**	**Required Software**
Web Browser on any platform	HTTP	Windows NT, Windows 2000	WGS	TCP/IP to non-CICS platforms. AS/400 Toolbox for Java to AS/400. CICS Transaction Gateway (CTG) to CICS platforms.	Windows NT, Windows 2000	WGS
		AIX	WGS		OS/2	WGS
		Solaris	WGS		AIX	WGS
		AS/400	HS400		Solaris	WGS
		OS/390 Unix System	HS		HP/UX	WGS
					IMS/VS	HS
					AS/400	HS400
					CICS for Windows NT	WGS
					CICS for AIX	WGS
					CICS for Solaris	WGS
					CICS for MVS/ESA	HS
					CICS for VSE/ESA	HS

Notes:
HS = VisualAge Generator Server for MVS, VSE, and VM
HS400 = VisualAge Generator Host Server for AS/400
WGS = VisualAge Generator Server for OS/2, Windows NT, AIX, HP-UX, and Solaris

You can use the following table to determine the platforms that are supported for your client usage. For example, if your client usage is a Java GUI Application, the supported platforms shown in the following table are: Windows 95, Windows 98, Windows NT, Windows 2000, OS/2, AIX, and Solaris.

Table 65. Usage and Platform Matrix

Platform	Usage								
	ST GUI	**Java GUI Application**	**C++ TUI**	**ITF**	**COBOL TUI**	**Java Servlet**	**Applet Web Server**	**Enterprise Java Server**	**Web Transaction Gateway**
Windows 95	x	x							
Windows 98	x	x							
Windows NT	x	x	x	x		x	x	x	x
Windows 2000	x	x	x	x		x	x	x	x
OS/2	x	x	x	x			x		
AIX		x	x			x	x	x	x
Solaris			x			x	x	x	x
HP/UX			x						
VM/ESA					x				
IMS					x				
AS/400					x	x	x	x	x
CICS for Windows NT			x						
CICS for OS/2					x				
CICS for AIX			x						
CICS for SOLARIS			x						
CICS for MVS/ESA					x				
CICS for VSE/ESA					x				
OS/390 Unix						x	x	x	x
SCO OpenServer			x						
Notes: x = Supported blank = Not available									

You can use the following table to determine the protocol options available from your client or 2nd tier server to your 2nd or 3rd tier server. For example, if your client is a Java GUI Application running on Windows 98 and your 2nd tier server is CICS for Windows NT, the available protocol shown in the following table is CICS CLIENT.

Table 66. From/To Protocol Matrix

From	To													
	Windows NT	CICS for Windows NT	OS/2	CICS for OS/2	AIX	CICS for AIX	Solaris	CICS for Solaris	HP/UX	VM/ESA	IMS	AS/400 server program (COBOL)	CICS for MVS/ESA	CICS for VSE/ESA
Windows 95*, Windows 98*	T,D	CC	T,D	CC	T,D	CC	T	CC	T	T	A	CA, ATJ	CC	CC
Windows NT, Windows 2000	T,D	CC	T,D	CC	T,D	CC	T	CC	T	T	A	CA, ATJ	CC	CC
OS/2	T,D	CC	T,D	CC	T,D	CC	T	CC	T	T	A	CA, ATJ	CC, L	CC
AIX	T,D	CC	T,D	CC	T,D	CC	T	CC	T	T		ATJ	CC	CC
Solaris	T	CC	T	CC	T	CC	T	CC	T	T		ATJ	CC	CC
HP-UX														
VM/ESA														
IMS														
AS/400												ATJ		
CICS**		CD		CD		CD		CD					CD	CD
OS/390 UNIX													CE	
SCO OpenServer	T	CC	T	CC	T	CC	T	CC	T	T			CC	CC
Web transaction gateway***	T	CTG	T	CTG	T	CTG	T	CTG	T	T	ITOC	ATJ	CTG	CTG

Note:
blank = not available
A = APPC
ATJ = AS/400 Toolbox for Java. It is valid only when the related client is a Java Application, a Java servlet, an applet Web server, an enterprise Java server, or a Web transaction gateway.
CA = Client Access/400
CC = CICS client
CD = CICS DPL
CE = CICS EXCI
D = DCE
L = LU2
T = TCP/IP
CTG = CICS Transaction Gateway
ITOC = IMS TCP/IP OTMA Connection (IMS TOC).
* Platform support for clients only
** CICS for Windows NT, CICS for OS/2, CICS for AIX, CICS for Solaris, and CICS for MVS/ESA
*** On Windows NT, Windows 2000, OS/2, AIX, SOLARIS, VSE/ESA, AS/400, and OS/390 Unix System.

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

VisualAge Generator
Client/Server Communications Guide
Version 4.5

Publication No. SH23-0261-01

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

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

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

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

Please tell us how we can improve this book:

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

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

Name	Address
Company or Organization	
Phone No.	

Readers' Comments — We'd Like to Hear from You
SH23-0261-01

Cut or Fold
Along Line

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

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

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

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Information Development
Department G7IA / Bldg 062
P.O. Box 12195
Research Triangle Park, NC
27709-2195

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

SH23-0261-01

Cut or Fold
Along Line

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SH23-0261-01