

CA Common Services for z/OS

Administration Guide

Release 14.1.00



Second Edition

This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the "Documentation") is for your informational purposes only and is subject to change or withdrawal by CA at any time.

This Documentation may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Documentation is confidential and proprietary information of CA and may not be disclosed by you or used for any purpose other than as may be permitted in (i) a separate agreement between you and CA governing your use of the CA software to which the Documentation relates; or (ii) a separate confidentiality agreement between you and CA.

Notwithstanding the foregoing, if you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with "Restricted Rights." Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2012 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

CA Technologies Product References

This document references some of the following CA Technologies products:

- CA 1® Tape Management
- CA 7® Workload Automation
- CA 11™ Workload Automation Restart and Tracking
- CA ACF2™
- CA Allocate™ DASD Space and Placement
- CA Audit
- CA Automation Point
- CA Balancing
- CA Bundl®
- CA Database Analyzer™ for DB2 for z/OS
- CA Datacom®/AD
- CA Data Compressor™ for DB2 for z/OS
- CA DB2
- CA Deliver™
- CA Disk™ Backup and Restore
- CA Dispatch™
- CA Earl™
- CA Endeavor® Software Change Manager
- CA Fast Check® for DB2 for z/OS
- CA Fast Index® for DB2 for z/OS
- CA Fast Load for DB2 for z/OS
- CA Fast Recover® for DB2 for z/OS
- CA Fast Unload® for DB2 for z/OS
- CA IDMS™
- CA IDMB™/DB
- CA Insight™ Database Performance Monitor for DB2 for z/OS
- CA Index Expert™ for DB2 for z/OS
- CA JARS®
- CA JARS® Resource Accounting

- CA Jobtrac™ Job Management
- CA Log Analyzer™ for DB2 for z/OS
- CA Mainframe Software Manager™ (CA MSM)
- CA Merge/Modify™ for DB2 for z/OS
- CA MIA Tape Sharing
- CA MIC Message Sharing
- CA MICS® Resource Management
- CA MII Data Sharing
- CA MIM™ Resource Sharing
- CA NetMaster® File Transfer Management
- CA NetMaster® Network Automation
- CA NetMaster® Network Management for SNA
- CA NetMaster® Network Management for TCP/IP
- CA NetMaster® Network Operations for TCP/IP
- CA NetSpy™ Network Performance
- CA Network and Systems Management
- CA NSM System Status Manager
- CA OPS/MVS® Event Management and Automation
- CA Partition Expert™ for DB2 for z/OS
- CA Plan Analyzer® for DB2 for z/OS
- CA Quick Copy for DB2 for z/OS
- CA Rapid Reorg® for DB2 for z/OS
- CA RC/Extract™ for DB2 for z/OS
- CA RC/Migrator™ for DB2 for z/OS
- CA RC/Query® for DB2 for z/OS
- CA RC/Secure™ for DB2 for z/OS
- CA RC/Update™ for DB2 for z/OS
- CA Recovery Analyzer™ for DB2 for z/OS
- CA Roscoe®
- CA Scheduler® Job Management
- CA SYSVIEW® Performance Management
- CA Service Desk (Service Desk)
- CA Spool™ Enterprise Print Management

- CA SQL Ease® for DB2 for z/OS
- CA SYSVIEW® Performance Management
- CA TCPAccess™ Communications Server for z/OS
- CA TLMS Tape Management
- CA Top Secret®
- CA TPX™ Session Management for z/OS
- CA Value Pack for DB2
- CA Vantage™ Storage Resource Manager
- CA View®
- CA XCOM™
- CA Workload Control Center

Contact CA

Before contacting CA Technologies Support, do the following:

- Verify that TCP/IP is active and functional. Issue the following z/OS console command:

```
DISPLAY TCPIP
```

- Verify that CAICCI is active and functional. Issue the following z/OS console command:

```
DISPLAY A, name
```

name

Identifies the name of the CAICCI started task.

- Verify that the desired CA Service Desk server is accessible. Logon to TSO on the system where CAICCI is active and issue the following TSO command:

```
PING hostname
```

hostname

Identifies the host name assigned to the server where CA Service Desk is running.

- Verify that the CAISDI/soap server address space is active. Issue the following z/OS console command:

```
DISPLAY A, name
```

name

Identifies the name of the CAISDI/soap started task.

- Verify that CA Service Desk is started and functional on the desired server.
- Review the SYSLOG for any messages that might provide an indication of a problem condition.
- For problems with CA products that use CAISDI/elmds, make sure the CAISDI/elmds address space is active. Issue the following z/OS console command:

```
DISPLAY A, cdyfapi
```

cdyfapi

Identifies the name of the CAISDI/elmds started task (or job).

- For problems with CA Technologies products that use the standalone CAISDI/med address space (not elmds), make sure the standalone CAISDI/med is active. Issue the following z/OS console command:

```
DISPLAY A, medname
```

medname

Identifies the name of the standalone CAISDI/med started task.

- For problems with CA Technologies products that use the standalone CAISDI/els address space (not elmds), make sure the standalone CAISDI/els interface has been properly initialized. Issue the following z/OS console command:

```
S CASDIELS, CMD=ELSLIST
```

This produces a report of all products currently defined to the standalone CAISDI/els interface. This report shows the event status of each product's events. Each product begins on a new page with the product and its current status being displayed in the report headings. Make sure the product in question appears in the report and that "Status: Enabled" is displayed for that product.

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At <http://ca.com/support>, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at <http://ca.com/docs>.

Documentation Changes

The following documentation updates have been made since the last release of this documentation:

- [CAMODID](#) (see page 133)—Added chapter that describes the new CAMODID TSO Command Utility is added.
- [Additional Considerations for Logs](#) (see page 227)—Added note about CA LServ and multi-volume log support.
- [IF/ENDIF Logic Statements and System Symbols](#) (see page 100)—Added this topic to note the feature released in PTF RO52581
- [Common Address Space Shell](#) (see page 335)—Added this chapter to note the feature released in PTF RO52583.
- [CAIENF/CICS Troubleshooting Checklist](#) (see page 342)—Removed J152DCM2 from question number 3 because it does not apply to r14.1.

Contents

Chapter 1: Introduction	19
General Description	19
Functionality and Services.....	19
Manager/Agent Technology	20
Components	21
Management Services.....	21
Services Provided by CA Common Services for z/OS	23
Chapter 2: Event Management	25
File Structure Considerations on z/OS	25
zSeries File System Architecture	26
Usage Considerations on z/OS	27
Clean Up the Log Files	27
Store and Forward (Optional)	27
Event Management Messaging Examples.....	28
Write Messages Using cawto	28
Use cawto from a Batch Program	29
Integrating with CA OPS/MVS Event Management and Automation	30
Reply to Messages Using cawtor.....	31
Unicenter Browser Interface	31
Toolbar	32
Get Started	32
Access Service Manager	33
Implementation Process	33
Date and Time Controls for Automated Event Processing.....	34
Calendar Administration	34
Calendar Sharing	34
Message Traps and Actions.....	34
Event Management Policy Activation	49
Message Traffic Monitoring	49
Console Customization.....	51
SNMP Facilities.....	52
Security	58
Protecting Event Management under HFS Security	59
Web Server Requirements	59
NSMJSERV (Java Server) Requirements	60

NSMEMSTR Requirements.....	61
Java GUI Requirements	61
Event Management Utilities.....	62
How the Event Management Utilities Process Works	63
Prerequisites	64
Configuration Statements	64
Event Management Utilities Drivers	68
Run the Utilities as an OMVS Process	69
Route z/OS Console Messages to Event Management Utilities.....	70
Processing Multi-line WTO Messages	71
Issue Console Commands Using CAconsole	72

Chapter 3: Agent Technology **75**

Overview	75
How an Agent Works.....	76
Agent Protocol	76
Managed Objects	76
Agent Status	77
Components	77
Service Control Manager.....	77
Services	78
Distributed Services Bus.....	78
SNMP Administrator	79
Installation Considerations.....	79
AWADMIN Account.....	79
Security Requirements.....	80
Configure the aws_admin Service	80
Environment File (ENVFILE).....	81
Start Agent Technology	81
Operation	81
Online Commands.....	82
Mainframe Agent Installation Considerations	82
Agent MIB Files	82
Communicate Between Agents and Services.....	83
The Discovery Process.....	83
Performance Considerations.....	86
Adequate CPU	86
The Object Store Database	87
Determine the Best Values for the Three Caches	88
Customize Cache Sizes	89
Configuration.....	89

Agent Level.....	90
Event Manager Level.....	90
Turn off Statistics Collection	90
Integration with CA OPS/MVS Event Management and Automation	91

Chapter 4: Resource Initialization Manager 93

Overview	93
Features.....	93
How the Process Works	94
Parameter Statements	94
Auto Commands.....	95
Verification of Initialization	95
Operation	96
Run CAIRIM as a Started Task	96
Run CAIRIM Under the Master Subsystem	97
Run CAIRIM as a Batch Job.....	97
Automate the Startup of CAIRIM at IPL	98
CAIRIM Auto Commands Member	98
Error Handling	99
IF/ENDIF Logic Statements and System Symbols	100
Verification Utilities.....	104
Operation	104
CAIRIMU Utility	104
CAISMFU Utility.....	106
CAISUBU Utility	107
CA LMP	108
How CA LMP Works	108
Product Execution Key Control Statements	109
Add Product Execution Keys	110
SITEID Type LMP Keys	111
Load Product Execution Keys	114
Create a Customized CAS9 Procedure for Loading Product Execution Keys.....	115
Load New LMP Keys	115
Reduce CA LMP Console Messages.....	116
Emergency Key Generator	117
System Authorization Facility Support.....	120
CA LMP Errors	121
LMP Key Check Invocation Exit	121
LMP CA Product Usage Registration	122
CAISSF.....	125
Reporting Licensed Registered Product Usage	125

Record SMF Type 89, Subtype 2 Records	126
Generate the Software Product Registration Report	127
CAS9INIT CAIRIM Initialization Routine	130
CAS9INIT Parameter Statement	131
How CAS9INIT Initializes CAISSF When Security Product Is Not Yet Active	132
CAISSF RACF Class Table Parameters	132
CAMODID	133
Using CAMODID	134
Functions (CAMODID)	135
Operands (CAMODID)	135
Invoke CAMODID	136
JCL to run CAMODID	136

Chapter 5: Event Notification Facility 137

Overview	137
Features	138
Operation	138
View Event Names	139
Run CAIENF as a Started Task	139
Run CAIENF under the Master Subsystem	139
Stop CAIENF	140
Restart CAIENF	140
CAIENF Start Options	141
CAIENF Auto Commands Member	141
Configure DCM Statements	142
Replace a DCM	143
Handling ENF Database Full Conditions	144
Control Options Sharing	146
Archive Events	147
Restore Events	149
CAIENF/CICS Operation	150
How CAIENF/CICS Checks for Intercept Modules	151
Configure CAIENF/CICS to Install Intercepts Automatically	151
Configure CAIENF/CICS to Install Intercepts Automatically in Specific CICS Regions	152
Configure CAIENF/CICS to Install Intercepts Automatically for Specific CICS Releases	152
Install CAIENF/CICS Intercepts Manually	153
Activation of CA Technologies Products (CAIENF/CICS)	153
CAIENF/CICS SPAWN Communications Facility Operation	154
How CAIENF/CICS SPAWN Checks for Intercept Modules	154
Configure CAIENF/CICS SPAWN to Install Intercepts Automatically	155
Install CAIENF/CICS SPAWN Intercepts Manually	155

CAIENF/DB2 Operation	155
Configure CAIENF/DB2 to Install Intercepts.....	156
Disable or Enable the Installation of CAIENF/DB2 Intercepts	156
Activation of CA Technologies Products (CAIENF/DB2)	156
CAIENF/USS Operation	157
Reinitialize CAIENF/USS.....	157
Reinitialize CAIENF/USS by Restarting CAIENF	158
SNMP Monitor.....	158
Start the SNMP Monitor	159
Stop the SNMP Monitor	159
Trace and Debug	159
Environment Variables.....	160
Configuration File.....	162

Chapter 6: Common Communications Interface 163

Overview	163
Features.....	164
Configuration.....	165
Communications Protocols	167
How CAICCI Routes Control Information	168
CAICCI on PCs	169
Activate CAICCI.....	170
Customize the CAICCI Service.....	170
TCP/IP Configurations	170
Client-Server.....	171
TCP/IP Gateway.....	171
Assign TCP/IP Ports	172
Assign TCP/IP Ports to Multiple Instances of a CAICCI Server	172
Example: Configuration with Two z/OS Systems and Multiple PCs	173
Customize the CAICCI TCP/IP PROC Names	174
Configure CAICCI on Windows Running CA Technologies Products	176
Issue CAICCI TCP/IP Gateway Commands	178
SNA Configurations	186
Define VTAM Resources.....	187
Configure CAICCI for SNA	188
z/OS Parallel Sysplex Configurations.....	190
Configure CAICCI for XCF.....	191
Configure CAICCI for XES.....	192
CAICCI Generic Resources	193
Configure CAICCI Generic Resources	194
CAICCI SPAWN Facility.....	194

Configure CAICCI SPAWN Facility.....	195
Assured Delivery.....	195
Define the Database.....	195
Activate Assured Delivery	197
View the Database Contents.....	197
Cross Platform Scheduling.....	200
XPS Client	200
CAICCI Connections	200
z/OS Connections	201

Chapter 7: Windows to Mainframe: Common Communications Interface 203

PC-to-Mainframe Client-Server Configuration on Windows.....	203
TCP/IP.....	203
Install CAICCI-PC.....	204
Configurator	204
Configure CAICCI for TCP/IP	205
TCP/IP Tab	206
SSL Tab	207
Test the Configuration.....	210
Trace a Communications Problem	210
Trace Tab.....	211

Chapter 8: CA-L-Serv 213

CA-L-Serv Operation.....	214
Kernel Server Relationship	214
The Kernel	214
CA-L-Serv Configuration	217
Choose System and Subsystem Names.....	217
Identify CA-L-Serv to Client Applications	218
Provide Operating Values for CA-L-Serv.....	218
Define Message Tables for CA-L-Serv	219
Activate the Servers	221
Start CA-L-Serv Servers	221
Define SQL Relational Tables for CA-L-Serv	222
Multiple System Environment Configuration.....	223
Deactivate CA-L-Serv Servers and Tasks	225
Collect Data in the CA-L-Serv Logs	226
Define Log Files	228
Log File Operations	230
Obtain Dumps	231
Display Information about CA-L-Serv	232

Configure the File Server	233
Start and Stop the File Server	233
Define Host and Remote Servers	235
Configure Communications in a Multiple-System Environment	236
Adjust the Size of Data Buffers.....	238
Define Managed Files.....	239
Place Files under the Management of CA-L-Serv	240
Buffer Pools.....	241
Performance Adjustments	245
File Groups	246
Set File Access	248
Propagate ENQ Requests	248
Protect Your Data Sets	249
Manage the File Server	249
Display Information about the File Server	249
Respond to System Outages	250
Maintain Managed Files	251
Set File Availability	252
The LDMAMS Utility	252
Back Up and Restoring Files	253
Delete the Contents of Files.....	256
Compress Files	256
Archive Files and File Groups	257
Use the Communications Server	260
Communication Protocols.....	260
Use of Multiple Communication Services and Protocols	261
Set Up the Communications Server	262
Start and Stop the Communications Server	262
Display Information about the Communications Server	263
Configure XCF Communication	263
Start the Communications Server with XCF	263
Start the Communications Server with XCF and VTAM	264
Configure Data Transmission Values.....	264
Respond to XCF Communication Problems.....	267
Configure VTAM Communication.....	267
Start the Communications Server with VTAM	268
Start the Communications Server with XCF and VTAM	268
Configure Data Transmission Values.....	270
Respond to VTAM Communication Problems.....	273

Chapter 9: CA Global SubSystem

275

Overview	276
Components	277
Run CA-GSS.....	277
Stop CA-GSS.....	278
Initialization Parameters	278
IMODs.....	282
Service Routines	282
Server IMODs	284
Special Purpose IMODs	288
System Security	289
User IDs	289
How CA-GSS Uses User IDs.....	289
How IMODs Execute.....	291
IMOD Facilities	292
Data Set Security	292
IMOD Naming Conventions.....	292
Name Prefixing.....	293
IMOD Editor	293
Macros	294
Data Integrity	295
IMOD Source Recovery	295
REXX Language in CA-GSS.....	296
Binary Conversion in REXX	298
ADDRESS Environments	299
Extended Return Codes.....	303
Program Stacks.....	304
Work Stacks.....	308
GoalNet	308
ILOG Files.....	309
How IMOD Variables Are Accessed.....	312
Predefined Variables	312
Global Variables	314
External Subroutines	314
Compiler Directives	315
User-defined Functions and ADDRESS Environments	315
User ID Routine	316
Data Stack Routine	316
Variable Access Routine	316
SAY Instruction Routine	316
Function Arguments.....	316

Function Return Codes.....	316
ADDRESS Environment Arguments	317
ADDRESS Environment Return Codes	317
Coding Requirements.....	317
Example: Function.....	319
Example: ADDRESS Environment	319
How IMODs Are Created	320
Allocate New ISET Data Sets.....	321
Define an ISET to CA-GSS.....	321
Write an IMOD	322
Compile an IMOD	323
Load a Compiled IMOD	323
Test a Loaded IMOD	324
IMOD Execution Using SRVBATCH	326
Considerations	326
Execute a Compiled IMOD Using SRVBATCH	327
Example: IMOD for Processing Operator Commands	328
Package IMODs as a Load Module	330
Debugging Considerations	331
Back Up IMODs.....	332
Restore an ISET.....	332
Restore a Single IMOD	332
Migrate ISETs from Previous Releases	333
Batch Maintenance of ISETs and ILOGs.....	333

Chapter 10: Common Address Space Shell **335**

Overview	335
Security.....	336
Create a Common Address Space through a z/OS START command	336
Create a Common Address Space as a Batch Job.....	337
Run a Server Application in a Common Address Space Shell	337
Customize a Common Address Space Shell.....	337

Appendix A: Troubleshooting **339**

Collect Diagnostic Data	339
Interpret Diagnostic Data	341
Identify and Resolve the Problem	341
CAIENF/CICS Troubleshooting Checklist	342
CAIENF/DB2 Troubleshooting Checklist	343
CAIENF/USS Troubleshooting Checklist	344
CAICCI Troubleshooting Checklist	345

USS Environment Troubleshooting	346
Event Management Troubleshooting Checklist	348
Agent Technology Troubleshooting Checklist	351
CA4FIVP Options	353
CA TLC: Total License Care	353
Product Versions and Maintenance	354
Request Enhancements	354
Appendix B: CAIENF Batch Database Query and Administration	355
Index	357

Chapter 1: Introduction

This guide provides both conceptual and practical information about the various components that comprise CA Common Services for z/OS. It includes detailed explanations regarding their usage and their configuration.

Note: For information about commands, control options, and utilities, see the *Reference Guide*.

This section contains the following topics:

[General Description](#) (see page 19)

[Functionality and Services](#) (see page 19)

[Components](#) (see page 21)

General Description

CA Common Services for z/OS extends the choice of managing your enterprise from anywhere by providing a z/OS-hosted enterprise management service that is similar to the existing CA Common Services on platforms ranging from Windows to UNIX. It also contains all the essential components and functionality to enable integrated management of z/OS.

Functionality and Services

CA Common Services for z/OS enables you to do the following:

- Use existing CA Technologies management applications, such as CA 1 Tape Management, CA OPS/MVS Event Management and Automation, and CA Top Secret Security to manage your mainframe z/OS system as part of a heterogeneous enterprise.
- Manage emerging z/OS workloads such as web servers, Java applications, and UNIX applications.
- Achieve enterprise-wide, automated, high-level monitoring and management of critical resources using sophisticated manager/agent technology.
- Facilitate communication between numerous CA Technologies products, such as CA Endeavor Software Change Manager and CA Insight Database Performance Monitor for DB2 for z/OS Performance Monitor, using the CA-GSS interface.

- Provide CA Technologies products with the file management, cross-system communications, and structured query language (SQL) table management services of CA-L-Serv.
- Run existing z/OS agents using the Agent Technology infrastructure.

Manager/Agent Technology

CA Common Services for z/OS employs manager/agent technology to facilitate comprehensive enterprise management. Using this technology, the components that produce management data and take action on behalf of managers are architecturally separate from the components that use management information, control management actions, and delegate management authority.

Agents

An agent is an application that supports enterprise management. An agent typically resides on a managed computer and provides information to a management application with a simplified and standardized view of monitored data.

Agents let you achieve enterprise-wide, automated, high-level monitoring and management of critical resources including hardware, software applications, and network devices. Agents provide the service of gathering information about your information technology (IT) infrastructure through remote access monitoring and control of resources.

An important feature of agents is their ability to "instrument" a resource so that specific information about that resource can be gathered and the resource managed. For example, suppose you are interested in just a few specific values in a large database maintained by a manufacturing application. An agent can be written that monitors those values and advises you when the values meet certain criteria.

The sophisticated Agent Technology of CA Common Services for z/OS takes the idea of instrumenting a device to a higher level—it is possible to instrument practically any resource in your entire IT infrastructure. CA has several pre-packaged agents for the various CA Technologies platforms.

Managers

Of the various components that contribute to the comprehensive management capabilities of CA Common Services for z/OS, the agent manager is most essential. Through the action of various agent runtime components, Simple Network Management Protocol (SNMP) traps and messages regarding the status of critical resources can be routed to Event Management.

Agent management entails *event-driven* processing whereby all activities are triggered in response to events, which can originate from network polling, timers, traps, and state transitions. Event-driven processing enables efficient use of your resources by ensuring that data is processed only when necessary.

Note: For a comprehensive discussion of agents and managers, see the CA NSM documentation.

Components

CA Common Services for z/OS consists of a management layer and a common z/OS services layer. The management layer provides a web-based real-world Interface, and monitoring of application and system events. The common z/OS services layer includes a suite of industry standard integration and distributed processing services to unify software applications.

Management Services

Management services enable integrated administration of all IT resources in your enterprise, including network devices, databases, business applications for desktop systems and mainframes, and all servers between.

Event Management

Event Management is a collection of management components that employ a single, easy-to-use graphical user interface (GUI) to monitor and administer different kinds of asynchronous events including SNMP traps, application events, and system events. The GUI included with Event Management on z/OS provides access to the Event Management Console, the management of Message Action records, and the management of calendars.

CA Common Services for z/OS Event Management augments z/OS automation solutions, such as CA OPS/MVS Event Management and Automation, by providing built-in access to a wide range of distributed events and by allowing actions to be triggered on any CA Common Services equipped platform. CA Common Services for z/OS provides event correlation and event processing and is fully integrated with CA Common Services event management facilities on other platforms.

You can define specific Event Management policy to do the following:

- Respond to messages
- Suppress messages
- Issue CA Common Services for z/OS commands
- Start other programs or scripts

- Send information to a network management or automation application such as CA OPS/MVS Event Management and Automation
- Forward messages to other managed platforms
- Issue commands to be executed on other platforms
- Interpret the results of any action to decide if additional actions are warranted

Event Management can be configured to process messages on individual servers and redirect them to a central server or other servers and, by extension, their consoles. Event Management makes it easy to collect related messages throughout the network for display at a single location or send them to multiple locations, as needed.

The Event Console Log GUI window enables you to monitor system events as they occur. All running programs and user processes can direct inquiries and informative messages to this facility. It provides a complete view of processes across the network.

Calendars

Calendars make it possible to determine a course of action based on *when* an event occurs. The event not only meets the general criteria; it also meets the date, day, and time criteria established through calendar profiles. The primary function of a calendar can be identified in a naming scheme. CA Common Services for z/OS provides facilities to define as many calendars as you require to meet your needs and to store them for easy reference.

Note: The use of calendars with Event Management is optional.

Agent Technology

The Agent Technology infrastructure enables the use of agents for the z/OS environment. The agents report to agent managers, which monitor and report the status of your resources and applications. It supports existing prepackaged z/OS agents such as the z/OS system agent, CA-IDMS agent, DB2 agent, and CICS agent, as well as other agents created to CA Common Services specifications.

Agent Technology supports a wide range of platforms and is deployable in traditional client/server, internet, and intranet environments. This versatility enables enterprise-wide monitoring and management of z/OS elements and further enhances the ability of z/OS environments to participate in a true heterogeneous network.

Services Provided by CA Common Services for z/OS

CA Common Services for z/OS provides the following services:

- **CAIRIM**—Prepares your operating system environment for all of your CA Technologies applications and then starts them. It is the common driver for a collection of dynamic initialization routines that eliminate the need for user SVCs, SMF exits, subsystems, and other installation requirements commonly encountered when installing systems applications.
- **CAICCI**—Provides CA Technologies enterprise applications with a common communications software layer that insulates the applications from dealing with protocol specifics, error recovery, and system connection establishment.
- **CAIENF (Base, CICS, and DB2)**—Provides comprehensive operating system interfacing services to any of the CA Technologies z/OS applications, exploiting technologies such as relational database architectures, for the benefit of the entire product line. The level of integration is improved by enabling operating systems and CA Technologies software-generated event information to be driven through a standard interface, simplifying multiple product-to-product interfaces and associated maintenance that would otherwise be necessary.
- **CAIENF/CICS SPAWN**—Enables CA Technologies applications to start CICS units of work from outside the CICS region. This facility provides a layer that isolates the application software from CICS release dependencies.
- **Management of z/OS UNIX System Services (CAIENF/USS)**—Encapsulates and integrates the management of UNIX System Services applications on z/OS. This service enables management applications to process system events occurring in the z/OS UNIX System Services subsystem.
- **CAISDI**—Provides a set of services that open CA Service Desk requests from the z/OS environment. The requests can be opened directly by CA Technologies products or they can be opened on their behalf, depending upon the requirements of each specific product using the interface.
- **CAISSF**—Provides an external security mechanism for controlling and monitoring access to all system and application resource processes. CAISSF is already well integrated into many CA Technologies enterprise applications, and is used by other CA Common Services for z/OS services as well. It provides security services for user sign-in, resource access control, process use control, and recording and monitoring of violation activity.
- **CA LMP**—Provides a standardized and automated approach to the tracking of licensed software.
- **Earl Service**—Combines a user-friendly report-definition facility with the power of a comprehensive programming system. Earl Service allows you to modify and print the contents and layout of a predefined CA Technologies application report using English-like statements.

- **Easytrieve Service**—Provides easy-to-use information retrieval, sophisticated report writing, and comprehensive application development capabilities. The Easytrieve Service is a limited version of the generally available, full-featured product, which allows you to modify the contents of an Easytrieve application that is provided with another CA Technologies product. If you have CA Easytrieve already installed at your site, you do not need to install the Easytrieve Service from CA Common Services.
- **SRAM Service**—Allows the activation of several sorts concurrently, thereby simplifying the data and logic flow. The incoming data to the sort can be manipulated as desired by the user program in a high-level language without the need for special exit routines.
- **CA-C Runtime**—Insulates programs from system and release dependencies.
- **CA-L-Serv**—Provides the CA-L-Serv services that are used by CA Technologies products including CA Endeavor Software Change Manager, CA Bundl, CA Balancing, CA TPX Session Management for z/OS, and CA MIC Message Sharing. These services include centralized logging and messaging facilities, VSAM file management, cross-system communications, and SQL table management.
- **CA-GSS**—Allows various CA Technologies products to communicate easily, seamlessly, and reliably, thereby providing quick access to information from various sources. CA-GSS provides connectivity by using a collection of one or more REXX subroutines that are edited, compiled, and executed as a single program.
- **CA-XPS**—Enables cross-platform scheduling for CA Technologies products including CA 7 Workload Automation, CA Scheduler Job Management, and CA Jobtrac Job Management.
- **Examine**—Provides a tool for CA Technologies Support to collect and report on installed instances of CA Technologies products at a particular customer site. This service is useful for troubleshooting problems.
- **CA Health Checker Common Service** - Provides CA applications with a framework for invoking IBM Health Checker Health Check routines in a consistent and easy to implement manner. This allows CA products to add reporting capabilities or recommendations for optimizing product settings as real world experiences dictate. Optional product features that should be activated can also be pointed out. The Health Checker results are constant real-time reviews of product settings.

CA product health checks run under the IBM Health Checker for z/OS. To successfully register CA checks, the IBM Health Checker must be active. For more information about the IBM Health Checker - including set up and configuration tasks - see the *IBM Health Checker for z/OS User's Guide* appropriate for your release of z/OS.

Chapter 2: Event Management

Event Management facilities provide a focal point for integrated message management throughout your heterogeneous network. They can monitor and consolidate message activity from a variety of sources. They let you identify event messages that require special handling and initiate a list of actions specified for handling those events.

You can channel event messages from any node in your network to one or more monitoring nodes through the support of industry standard facilities. This makes it easy to centralize management of many servers, and ensure that important events are detected and routed to where they can be acted on most expeditiously.

For example, you can route message traffic to three different event servers by directing event and workload messages to the event manager for production control, security messages to the event manager for the security administrator, and problem messages to the event manager for the help desk administrator. Further, by filtering the messages that appear on each console, you can retrieve specific information about a particular node, user, or workstation.

This section contains the following topics:

[File Structure Considerations on z/OS](#) (see page 25)

[Usage Considerations on z/OS](#) (see page 27)

[Event Management Messaging Examples](#) (see page 28)

[Unicenter Browser Interface](#) (see page 31)

[Implementation Process](#) (see page 33)

[Protecting Event Management under HFS Security](#) (see page 59)

[Event Management Utilities](#) (see page 62)

File Structure Considerations on z/OS

Because of the interdependence of the Event Management zSeries File Systems and the ability to share the executable code across systems, the structure of the file systems and mount point decisions are important.

zSeries File System Architecture

The zSeries File System (zFS) address space must be running under z/OS UNIX.

Prior to installing, decide how you want your zFS to be structured. Event Management is designed to use a read-only (RO) zFS that can be shared across systems and a read-write (RW) zFS for each system running Event Management. Mount the RW directory under the RO directory. For example, you might mount RO zFS at /cai/nsmem and mount the RW zFS at /cai/nsmem/RW. (For Agent Technology, the reverse is true: the RO directory is under the RW directory. The Agent Technology zFS files are shown in the following example as well as the Event Management zFS files.)

A typical BPXPRMxx mount setup might use the following example statements:

```
MOUNT          FILESYSTEM( 'CAI.DIR.HFS' )
                MOUNTPOINT( '/cai' )
                TYPE(HFS)  MODE(RDWR)

/*                                                    */
MOUNT          FILESYSTEM( 'CAI.NSMEM.RO.CAIZFS' )
                MOUNTPOINT( '/cai/nsmem' )
                TYPE(ZFS)  MODE(READ)

/*                                                    */
MOUNT          FILESYSTEM( 'CAI.NSMEM.RW.CAIZFS' )
                MOUNTPOINT( '/cai/nsmem/RW' )
                PARM( 'AGGRFULL(90,5)' )
                TYPE(ZFS)  MODE(RDWR)

/*                                                    */
MOUNT          FILESYSTEM( 'CAI.AGENT.RW.CAIZFS' )
                MOUNTPOINT( '/cai/agent' )
                PARM( 'AGGRFULL(90,5)' )
                TYPE(ZFS)  MODE(RDWR)

/*                                                    */
MOUNT          FILESYSTEM( 'CAI.AGENT.RO.CAIZFS' )
                MOUNTPOINT( '/cai/agent/services/bin' )
                TYPE(ZFS)  MODE(READ)
```

Note the following in this example:

- The /cai directory is added to the root directory.
- During the installation, CAI.DIR.HFS is mounted. This is a small HFS just for directory entries.
- mkdir commands are issued for creating the '/cai/nsmem' and '/cai/agent' directories.
- During the installation and the post-installation fwsetup script execution, all zFS data sets are mounted RDWR.

Usage Considerations on z/OS

Event Management on z/OS can be thought of as the following processes:

- **logrdr** - Responsible for listening for messages from various sources: syslogd, CA Technologies, messages generated using the EM API, and messages routed by remote applications.
- **caiopr** - Responsible for creating the CA Technologies Event Management console from the various message sources. Store-and-forward is controlled by caiopr.
- **ca_calendar** - Responsible for determining if a specific calendar is active or inactive based on the current date and time. If a message action is assigned a calendar profile, then ca_calendar must be active for the action to be taken.
- **stardaemon** - Responsible for transmitting data from the z/OS computer to a remote Windows computer. When trying to connect using the Windows GUI, this process is responsible for doing the log on.
- **catrapd** - Responsible for listening for SNMP traps. Traps received are displayed on the CA Technologies Event Management console.

The z/OS version does not provide such components as workload or security, which are best left to existing technologies already present on the mainframe.

There are other general implementation considerations related to these processes as well.

Clean Up the Log Files

Caiopr creates a new log file for each day it is in operation. Over time, the number of files can take up more storage than desired.

To limit the number of logs, set the environment variable `CA_OPR_RETAIN_LOGS` to the maximum number of log files you wish to keep. This helps keep storage use to a minimum.

To set the variable, edit the file `/cai/nsmem/opr/scripts/envusr`. The default setting is '0', which keeps all logs.

Store and Forward (Optional)

The store and forward option lets you automatically forward messages to another host. If that host is unavailable, it can store the message until that host becomes available.

To enable store and forward during installation, set the UPDATE_SAF value to Y.

To enable store and forward after installation

1. Set and export the environment variables in file /cai/nsmem/opr/scripts/envusr:

```
CA_OPR_SAF=Y
CA_OPR_SAF_ROOT=$CAIGLBL0000/opr/saf
```

2. Ensure that the \$CAIGLBL0000/opr/saf directory exists. It should be a symbolic link to ../RW/saf.

3. Create a file oprsaf in the \$CAIGLBL0000/opr/config/<node> directory. The file can be empty. It will contain the word <active> when Store and Forward is activated.

The Store and Forward daemon is the oprsafd process. This program is located in the \$CAIGLBL0000/bin directory and is started by this command:

```
UNICNTRL START OPR
```

When issuing a cawto command and implementing Store and Forward, the CA_OPR_SAF and CA_OPR_SAF_ROOT environment variables must be set within the issuer's environment.

Event Management Messaging Examples

This section provides Event Management messaging examples.

Write Messages Using cawto

You can use the cawto utility to write messages to the local Event Management console or to a remote computer. You can also assign different attributes to the messages to make them more meaningful. The basic syntax of the command is as follows:

```
cawto      -a attribute (DEFAULT, BLINK, REVERSE)
           -c color    (DEFAULT(White), Red, Orange, Yellow, Green, Blue,
                       Pink, Purple)
           -g category (user defined)
           -k           (Place in held message area)
           -n node     (The node name of the machine, if omitted defaults to
                       local machine)
           -s source   (Identifies the user defined source to the message,
                       useful for message matching)
```

For example, to call cawto to the local computer, specify:

```
cawto This is my message
```

To call cawto to a remote computer, specify:

```
cawto -n node This is my message
```

Use cawto from a Batch Program

You can call cawto using the BPXBATCH utility from IBM using either of the following methods:

- Directly in the PARM
- Using a script that calls cawto

The STDENV DD is used to set environment variables. The use of '\$varname' is not supported in this type of DD so the STDENV file must contain hard coded PATH, LIBPATH, and so on.

Use the PARM Statement

Here is an example of calling cawto in the PARM statement:

```
//EXPAND EXEC PGM=BPXBATCH,REGION=0M,
// PARM='PGM /cai/nsmem/bin/cawto Hi from Batch'
//STDOUT DD PATH='/cai/nsmem/RW/batch.out',
// PATHOPTS=(OwRONLY,OCREAT,OTRUNC),
// PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP,SIROTH)
//STDERR DD PATH='/cai/nsmem/RW/batch.err',
// PATHOPTS=(OwRONLY,OCREAT,OTRUNC),
// PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP,SIROTH)
//STDENV DD PATH='/cai/nsmem/RW/ENVFILE',
// PATHOPTS=(ORDONLY)
/*
/*
```

The ENVFILE could appear as follows and include any other variable that may be required:

```
PATH=/cai/nsmem/bin:/usr/bin:/usr/sbin
LIBPATH=/cai/nsmem/lib:/usr/lib
```

Use a Script

Here is an example of calling cawto from within a script:

```
//EXPAND EXEC PGM=BPXBATCH,REGION=0M,
// PARM='PGM /cai/nsmem/bin/myscript Hi from a script file'
//STDOUT DD PATH='/cai/nsmem/Rw/batch.out',
// PATHOPTS=(OwRONLY,OCREAT,OTRUNC),
// PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP,SIROTH)
//STDERR DD PATH='/cai/nsmem/Rw/batch.err',
// PATHOPTS=(OwRONLY,OCREAT,OTRUNC),
// PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP,SIROTH)
//STDENV DD PATH='/cai/nsmem/Rw/ENVFILE',
// PATHOPTS=(ORDONLY)
/*
**
```

The file myscript could do the following:

```
#Change to our directory
cd /cai/nsmem
# Set the environment variables
. PROFILE
# Take the parm passed as the message to be sent
cawto $1
# Send a copy of the command to the console
logger message $1 sent
```

Integrating with CA OPS/MVS Event Management and Automation

This section provides examples of CA OPS/MVS Event Management and Automation rules for interfacing with Event Management.

Write a Message to a Remote Node

Following is an example of writing a message to a remote node. In this example, the message is prefixed with HELPDESK. This could be used as a keyword on the remote machine to take an action, such as opening a help desk issue based on the information passed in the message.

```
)CMD USSWT061
)PROC
Address USS
"WTO ",
"TEXT('HELPDESK Open Help Desk issue PAYROLL processing abended') ",
" node(helpdesk) color(orange) attribute(reverse)"
return accept
```

Write a Message to a Remote CA Technologies Computer with an Active Web Server

Following is an example of writing a message to a remote CA Technologies computer with an active web server:

```
)USS IMW3536I
)PROC
/* Inform a remote Unicenter NSM that a IBM HTTPD server is active */
MSGTXT = 'OPSAUT01 IBM HTTPD server initialized at ' TIME()
Address USS
"WTO TEXT('"MSGTTEXT"') NODE(NSMNODE) "
end
return "Normal"
```

Issue a Command on a Remote CA NSM

Following is an example of issuing a command on a remote CA NSM:

```
)USS SENDCMD
)PROC
/* Execute a command on a remote machine (Unix, z/OS, or Windows) */
CMD = 'backup logfile '
Address USS
"CMD COMMAND('"CMD"') COLOR(blue) NODE(AIX1)"
end
return "Normal"
```

Reply to Messages Using cawtor

After you send a message, you can request a reply using the cawtor utility. This utility works similar to a WTOR message on the system console in that a reply ID is assigned to the message and the reply can be relayed back to the caller.

For example, assume you have a process that requires an action to be taken on a remote machine and you need to wait for that action to be completed before continuing. Using cawtor, the processing will hold until a response to the message is received. The response can be manual or automated on the remote machine.

Unicenter Browser Interface

CA Common Services management services are accessible through a GUI that offers a highly visual approach to monitoring and administering different kinds of asynchronous events including Simple Network Management Protocol (SNMP) traps, application events, and system events that occur throughout your enterprise.

The GUI uses Java, hypertext markup language (HTML), and virtual reality markup language (VRML) technology for web browser access. Web access to the management services is optionally secured through the Secure Sockets Layer (SSL), yet it provides for open and extensible management capabilities. From the browser interface, you can integrate many mainframe and distributed applications, including 3270 sessions, through Host On-Demand or equivalent service.

The browser interface provides all of the functionality of the Event Management components in an explorer-like manner. The left pane displays a tree view of your enterprise with detailed information on the right pane.

Note: For more information about Unicenter Browser Interface, see the online help.

Toolbar

The toolbar in Unicenter Browser Interface comprises two groups of buttons:

- **Toolbar Topic buttons**—Let you access the following CA Common Services for z/OS applications and exhibit their contents in the left pane tree-view:
 - Enterprise Management, which includes Event Management and Calendar Management
 - Links to useful web sites
 - Service Manager, which includes clients and servers
- **Display buttons**—Control the appearance of icons that represent the contents of the tree in the right pane (for example, changing the size of the icons).

Get Started

The Unicenter Browser Interface lets you access management services such as Event Management.

To get started with the CA z/OS Event Management Browser Interface

1. Start a web browser session with the computer running the web server used by CA Common Services for z/OS.

The Unicenter Browser Interface page appears.

2. Click a launch link to download the Unicenter Browser Interface, and wait for the Unicenter Explorer Logon dialog to open.

The Unicenter Explorer Logon dialog opens.

3. Specify your mainframe logon name and password, and click OK.
The Unicenter NSM window opens.
4. Click the plus sign for Enterprise Management on the left pane, and then drill down to reveal the available Enterprise Management services for the selected z/OS computer.
5. Explore the services. While you are exploring, you can use the following features:
 - Right-click an item to display its context menu that provides access to additional information and serves as an alternate method of reloading contents.
 - Use the History drop-down list box (directly below the toolbar) to revisit a previously accessed view.

Access Service Manager

The Service Manager lets you view the GUI clients and persistent application servers.

To access Service Manager

1. Click the Service Manager icon from the toolbar.
GUI Clients and Persistent Application Servers appear in the tree view.
2. Select a GUI client or persistent application server from the Service Manager tree structure.
The contents of the selected node appear in the right pane.

Implementation Process

The successful implementation of Event Management involves the following processes:

- Establishing date and time controls for automated event processing
- Trapping important event messages and assigning actions
- Putting the Event Management policy into effect
- Monitoring message traffic
- Customizing your console
- Using SNMP to monitor activity
- Maintaining security

Date and Time Controls for Automated Event Processing

Determining a course of action based on when an event occurs can be critical to the proper handling of the event. You define the "when" criterion using calendar profiles.

Note: The use of calendars with Event Management is optional. You can proceed to [Message Traps and Actions](#) (see page 34) if you are not setting date and time controls now.

CA Common Services for z/OS provides facilities to define as many calendars as you require to meet your needs, and to store them for easy reference.

You define calendars using the Enterprise Management GUI or the cautil command line interface. The GUI has a significant advantage over command line entry because it simplifies date and time specifications.

Note: For more information about calendar administration, see Enterprise Management in the online help.

Calendar Administration

All users authorized to administer calendars must:

- Have their userid added to the CAL_AUTH list in the \$CAIGLBL0000/secopts file.
- Meet the general security requirements listed for Event Management. For more information, see Plan the Installation in the *Installation Guide*.

Calendar Sharing

The CA Common Services for z/OS calendar object is a common object and thus is available for use by any of the Enterprise Management functions. We recommend that you identify the primary function of your calendar in a naming scheme.

You can create calendars for use by any of the Enterprise Management functions. For example, you can use a single holidays calendar for your company whenever company holiday dates need to be considered.

Message Traps and Actions

Through Event Management message record and message action profiles, you can identify the messages that are important to your operation and define the special processing to be performed automatically whenever CA Common Services for z/OS encounters these messages.

Event messages are generated by the following:

- CA Common Services for z/OS components
- System components
- Utilities (such as cawto)
- User programs

All of these messages are sent to an event daemon for processing. The default processing, in the absence of any policy, is to write the messages to the Event Console log file.

To define a message processing policy to filter the messages received by the Event daemon and define specific actions to be taken, you begin by defining message records that describe which messages need processing.

A message record defines the matching criteria. Each field in the event message has a corresponding field in the message record. Message record fields may contain wildcards to match a wider range of event messages.

A message record has associated message action records that describe what action to take when a message matches the message record.

Note: For step-by-step procedures for defining message records and actions, see Enterprise Management in the online help.

Message Sources

Messages can be directed to Event Management from a variety of sources:

- The cawto command sends a message to the Event Console.
- The cawtor command sends a message to the Event Console and waits for a reply. The message appears in the held messages pane and will not be deleted until the operator replies.
- The oprcmd command sends a request to execute a command to the designated target computers.
- The careply command replies to a message held by the Event Console.
- Enterprise Management components such as Workload Management, Security Management, and File Management generate messages directly to the Event Console.
- SNMP event messages and traps are routed to the node identified by the caiopr daemon as the provider for the Event Management service through the CA trap daemon, catrapd.

- On z/OS UNIX platforms, messages from the syslog daemon are routed through the syslog daemon to the Event Console. Messages issued through the logger utility are included as they also use the syslog daemon. These messages may have originated on a platform not running CA Common Services for z/OS. (Messages issued by CA Common Services for z/OS components are sent directly to the Event Console, not through syslog, in order to take advantage of message matching fields.)
- Agent Technology policies issue messages to the Event Console.

Note: For more information about the `cawto`, `cawtor`, `opr cmd`, `careply`, and `catrapd` commands, see Event Management Commands in the *Reference Guide*.

Messages That Require Special Handling

You identify messages that require special handling by creating message records. You then specify what those special handling requirements are by creating message actions that are associated with a particular message record.

Defined message records and message actions become a message handling policy that identifies the events that have special handling requirements and the actions that must be performed when they are detected.

Message Records

Event Management provides two categories of message records to identify important events:

Message

Represents the output text string received by Event Management and displayed on the Event Console.

Command

Represents the text string entered by someone operating the Event Console. (You can enter commands at the command field of the console, use customized buttons to automatically issue commands, or enter these commands as command line arguments provided to the `opr cmd` command.)

z/OS UNIX command output can be used as a source of text that you can substitute into the message text in database message records during the message matching process.

For example, the string "pwd" in the database record message text field causes the current directory to be inserted into the message text.

Message Actions

Message actions specify what Event Management should do when it detects a match between an input event message and a message record. Possible actions range from simply highlighting messages on the console display to replying to messages, opening problems, or executing commands or other programs.

For example, to ensure that a message catches the attention of the person responsible for monitoring the console, you can use either or both of these methods:

- Route the message to a held area of the console GUI where it remains until acknowledged by the console operator.
- Assign an attribute, such as highlighting or blinking, to make a message more noticeable on the Event Console.

How Message Activity Is Distributed

CA Common Services for z/OS lets you distribute message and action activity across multiple servers and their clients. This capability enables you to do the following:

- Create a central event log from which all servers can be monitored.
- Send selected messages to another server for processing.
- Manage special functions such as security or tape management on dedicated consoles.

Whenever the event database is loaded, it checks the Eval node of every message record against its own node name. If its node name matches the Eval node of the message record, the record and all associated message actions are read into memory. If there is no match, the message record is ignored. The set of message and message action records read into memory constitute the event policy for the current execution of the event daemon until the policy is reloaded from the CA Datacom/AD database by a restart or the opreload command.

How Remote Actions Are Processed

Through message record and action policies, you select a message based on content and then define the desired action. One of the message action capabilities is to instruct Event Management to perform an action on a specific (and potentially remote) computer. Actions such as sending the message to a remote computer or initiating a command on the remote computer are accomplished easily. For example, you can identify a network security event or a tape mount event for routing to an alternate computer simply by defining Event Management policies to that effect.

Actions that must be performed on remote nodes can be done synchronously. Action processing waits for the remote action to complete and return a completion code before proceeding to the next action. This means that the completion code received from the remote action can be tested as part of a message action and used to control subsequent processing. Remote actions are not attempted if the target node is known to be unreachable.

Message Routing to Remote Hosts

Routing messages to a remote computer through message record and action policies is most easily achieved by creating a message record that traps those events and a message action that uses the FORWARD action keyword and specifies the remote node to which you want the messages sent.

Because one message can have many message actions, you can send messages to multiple computers. You can specify the name of any computer that is currently defined for CAICCI remote communication.

Note: For more information about CAICCI configuration, see [Common Communications Interface](#) (see page 163).

On z/OS UNIX platforms, one source of event messages is the Berkeley syslog daemon, which can route messages to a CA Common Services for z/OS server for processing, even those originating from servers not running CA Common Services for z/OS.

Event Management takes advantage of the powerful messaging facilities provided by the syslog daemon to:

- Select from several priorities, levels, and facilities of messages.
- Route messages by level or priority to different devices.
- Route messages by level or priority to different hosts.
- Receive messages from other hosts for local display.

Note: For more information about configuring the Berkeley syslog daemon, see the Post-Installation Tasks for Event Management in the *Installation Guide*.

Note: If you use both the Berkeley syslog daemon and specific message action policies to reroute the same messages to the same remote computers, those messages will display twice on those remote computers because they were sent there twice, once by the Berkeley syslog daemon and again by Event Management.

Message Action Definitions

There are several types of actions you can use in any sequence or combination to automate the processing of an input or output message. Actions labeled “Windows only” can be sent to a Windows computer running CA NSM.

ALLOW

Permits execution of a command entered at the system console or issued using the oprcmd command.

Return Code: Return code from the executed command

ANNOTATE

(Windows only) Adds text as an annotation to the last console log record.

Note: For detailed information, see the online help.

AUTORPLY

Automatically answers a message on the Event Console that is waiting a reply. The Text field lets you specify the answer.

Return Code:

Zero—OK

Nonzero—Invalid reply, message not found, or other failure to deliver

Example: &1 reply

BANNER

Displays a scrolling ticker-tape style message in a separate window on the desktop of the node specified in the message action. That node must be a Windows server and must be a node that has the Event Management real-time components running on it.

The Text field specifies the text you want displayed.

COMMAND

Permits execution of a program or command. On UNIX platforms, this action is the same as UNIXCMD; you can use either term.

The Text field specifies text string for the command syntax. CA Common Services for z/OS interprets the first element of this text string as the name of a program, script, or batch file to execute. The remaining elements are passed as arguments.

On z/OS, COMMAND must name a shell script or a zSeries File System (zFS) executable that can be invoked using the "Spawn" system service.

You can use a number of command prefixes with the COMMAND action. For details on these prefixes, see the online help for the cautil MSGACTION statement.

CORRELATE

(Windows only) Sends a message based on the TEXT operand to the specified Distributed State Machine (DSM).

Note: For detailed information, see the online help.

DELAY

Causes message action processing to wait a specific amount of time before proceeding to the next action.

The Text field specifies the time.

Example: *hh:mm:ss*

DELKEEP

Deletes a live console message previously retained by a SENDKEEP action.

The Text field specifies the text that must be matched and then deleted. Wildcards are accepted.

Return Code:

Zero—Message found and deleted

Nonzero—Message not found

DISABLE

Marks the message record currently being processed as unavailable for selection. When disabled, an input or output message cannot be selected for automatic processing until the message record is enabled with the ENABLE option.

For example, a typical response to an invalid reply is that the reply is rejected and the original message reissued. If the errant reply comes from an incorrectly defined Event Management message action, the result is an infinite loop of invalid replies. You can address this loop exposure by using the DISABLE message action.

DISCARD

Prevents an unwanted message from being displayed at an operator console. The message is not placed in the Event Console Log.

ENABLE

Marks a message record that was previously disabled as available for selection.

EVALUATE

Initiates message selection based on the text supplied by the Text field. The text is itself matched against the message record as if it were a message received by Event Management. Then the actions associated with the matched text are performed. If a match is not found, EVALUATE functions identically to SENDOPER.

Return Code:

Zero—Match

Nonzero—No match

EXIT

Terminates message processing for this event. Further actions are skipped.

EXPORT

Sets the value of an environment variable in the process in which the actions are being performed and in the Event Manager.

The Text field specifies the variable and its value:

variablename=value

In the local environment of the Event Management daemon, *value* is assigned to this environment variable. Any variable you specify can be tested later through the TEST message action.

EXTERNAL

(Windows only) Permits execution of a user-defined function in a dynamic-link library (DLL).

Return Code: Return code from the executed function

FORWARD

Forwards a message to another server for further evaluation.

The Text field specifies the text to be forwarded. If no text value is specified, the original message is forwarded. You can use this action to let a central node process all the messages generated on a set of remote nodes. The FORWARD action sends an event to a remote node, and allows for further matching and action on the remote node.

GOTO

Chains non-consecutive actions together to enable branching to another action associated with the message. Optionally, you can base the branching on the return code from the last action to perform a conditional GOTO.

The Text field specifies the action sequence number to go to.

HILITE

Highlights the message when displayed on the Event Console.

The Text field specifies the text to highlight. If no value is specified, the original message is highlighted.

IGNORE

Indicates that all actions for this message are to be ignored and not performed.

PERMIT

Permits execution of a command that is either entered at the system console or issued using the `caiovr oprcmd` command.

PREVENT

Prevents the execution of a specific command entered at the Event Console. It also intercepts the `oprcmd` command to prevent execution of a specific command.

PROHIBIT

Prevents the execution of a specific command entered at the Event Console. It also intercepts the oprcmd command to prevent execution of a specific command.

SENDKEEP

Marks a message for persistent display in the held messages area of the Event Console. The console operator must acknowledge and delete the message.

The Text field specifies the text to display on the console. If no value is specified, the original message is displayed.

SENDOPER

Sends a message to the Event Console.

The Text field specifies the text to be sent to the console. If no value is specified, the original message is displayed. The SENDOPER keyword sends an action to a remote node where the event is logged and no further matching is possible.

SENDUSER

Sends the text specified by the Text field to the user who created the message.

Return Code: Return code from the UNIX write command

SUPPRESS

Prevents an unwanted message from being displayed on the Event Console but records the message in the Event Console Log.

The Text field specifies the message text to be suppressed on the Event Console.

UPDATEMAP

(Windows only) Updates the status of WorldView objects.

Note: For detailed information, see the online help.

UNIXCMD

Indicates that a command or program is to be executed. This action applies only to UNIX platforms and is the same as COMMAND.

The Text field specifies the UNIX command to be issued.

UNIXSH

Indicates that a UNIX shell script is to be executed or that you want to create a pipeline (for example, cmd1 | grep xx | yy). This action applies only to UNIX platforms.

The Text field specifies the UNIX commands or shell script to be executed.

WAITOPER

Sends the specified message to the CA Common Services for z/OS console and waits for a reply.

The Text field specifies the text to display. You can use the careply command to reply to the message.

Return Code:

Zero—OK

Nonzero—System error

WAKEUP

Waits until the specified time before executing the next action.

The Text field specifies the wake-up time.

Example: *hh:mm:ss*

WORLDVPOP

(Windows only) Populates the WorldView Common Database with objects.

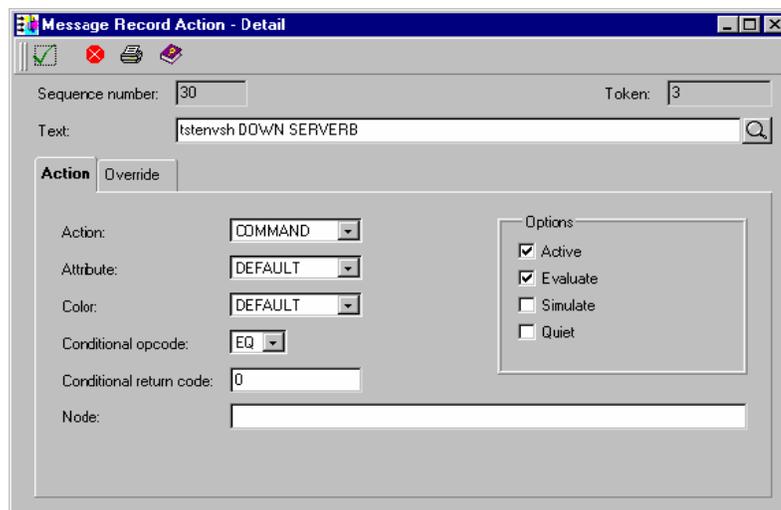
Note: For detailed information, see the online help.

Test Message Action

The TEST action is not yet implemented on UNIX. To perform this type of processing, you can execute a script as follows, using the action command to test the contents of a previously set variable.

```
#
# Args: 1 - The value to be tested for
#       2 - The variable to test
#
# Results: Equal - return 0.
#         False - return 1.
#
#
var='$'\`echo $2`
# Test if the variable has been defined first
if [ ! `eval echo $var` ]; then exit 1
fi
# It has been defined, test the value
if [ "$1" = `eval echo $var` ]; then
  exit 0
fi
# Not EQUAL exit false
exit 1
```

The following is an example of the Message Record Action - Detail screen defined to test if a variable SERVERB has the value of DOWN.



Message Action Restriction

Event Management lets you restrict the nodes and RUNIDs that are authorized to send the COMMAND, UNIXCMD, and UNIXSH message actions to your local host.

During the installation process, setup creates the actnode.prf configuration file. The file is in the `$CAIGLBL0000/opr/config/hostname` directory. It maintains the rules that specify how message action restriction is enforced based on the submitting node and RUNID. The file is owned by root, and only a UID of 0 can have write access to it. A rule in the file has the following format:

```
-n=nodename,runid,flag
```

nodename

Identifies the node from which the COMMAND, UNIXCMD or UNIXSH message action is initiated; it may contain a trailing generic mask character.

runid

Defines a RUNID to whom the rule applies; it may contain a trailing generic mask character.

flag

Defines *one* of the following values:

D—Disable (The feature is active to disallow the message action submitted by *runid* from *nodename*.)

E—Enable (Permit *runid* to submit the message action from *nodename*.)

W—Warn (Check the rule, but permit message action submission.)

You can update the `actnode.prf` configuration file at any time after installation by executing the `caevtsec` utility located in the `$CAIGLBL0000\bin` directory. The utility permits only the UID 0 user to maintain the file and preserve the file permissions. You can also maintain the file using a UNIX text editor.

Note: For more information about using the `caevtsec` utility, see the *Reference Guide*.

Example:

```
-n=*,*,E
```

This rule is in effect if, during installation, you elected not to activate message action restriction. The rule states that for all nodes and all RUNIDs, COMMAND, UNIXCMD, and UNIXSH message action submission is permitted.

Example:

```
-n=*,*,D
```

This rule is in effect if, during installation, you elected to activate message action restriction. The rule states that for all nodes and all RUNIDs, COMMAND, UNIXCMD, and UNIXSH message action submission is not permitted.

Example:

```
-n=*,*,E
-n=*,root,D
```

These rules enforce message action restriction on the root RUNID but permit all other RUNIDs to submit the message actions.

Example:

```
-n=*,*,E
-n=mars,*,D
-n=*,root,W
```

These rules permit all RUNIDs to submit the message actions unless the request comes from the `mars` node. In that case, message action restriction is enforced for all RUNIDs. The last rule sets a warning for the root RUNID if it comes from a node other than `mars`.

Event Management scans the entire configuration file for a best match and uses that rule. It uses the node field as a high-level qualifier when searching for a best match. In the example, any request coming from the mars node uses the “disallow” rule. The root user only triggers the warning rule if the request comes from a node other than mars.

Environment Variables for Messages and Actions

When Event Management invokes a command or script as a result of a COMMAND or UNIXCMD action, the new process is created with a list of new environment variables that contain information about the event that you may find useful. Programs, Windows BAT files, or UNIX shell scripts can reference these variables, but the variables cannot be altered.

The following table lists the variables and their descriptions:

Variable	Description
EVENT_CATEGORY	CATEGORY field from the event
EVENT_DATEGEN	Date (<i>yyyy/mm/dd</i>) the event was generated
EVENT_DEVICE	Device associated with the event
EVENT_JOBNAME	Event job name (if from a Workload job)
EVENT_JOBNO	Event job number (if from a Workload job)
EVENT_JOBQUAL	Event job qualifier (if from a Workload job)
EVENT_JOBSET	Event jobset (if from a Workload job)
EVENT_LOGRECID	Identification (ID) of the last record logged when the command is invoked (possibly the current event if it is not suppressed)
EVENT_MSGNUM	MESSAGE NUMBER field from the event
EVENT_NODEID	Node of origin associated with the event
EVENT_OPUSER	User name under which Enterprise Management is processing this message action
EVENT_PID	ID (decimal) of the process that generated the event
EVENT_PROGRAM	Program that generated the event (for example, cawto.exe)
EVENT_REPLID	Reply ID returned if the event was write-to-operator with reply (WTOR)
EVENT_SEQNO	Sequence number of the action that invoked this command
EVENT_SEVERITY	Event severity
EVENT_SOURCE	Event source
EVENT_STATION	Event station

Variable	Description
EVENT_TAG	Platform tag associated with the event (WNT, HPUNIX, and so on)
EVENT_TEXT	Full text of the event
EVENT_TIMEGEN	Time (<i>hh:mm:ss</i>) the event was generated
EVENT_TIME8	Time (<i>hh:mm:ss</i>) the command was invoked
EVENT_TOKEN	Token number of message record that matched this action
EVENT_TYPE	Type of event: MSG, CMD, REPLY, or WTOR
EVENT_UDATA	User data (value of the CA_UDATA environment variable when the event was generated)
EVENT_USERID	User of origin associated with the event
EVENT_YYYYMMDD	Date the command was invoked

Example:

The command file in the following submitted job definition references variables:

```
define msgrec msgid=* scantext=*error*
define msgact name=(*,100)
    action=commandtext=logerr.cmd
```

The logerr.cmd file contains:

```
@echo off

Rem This command file is invoked from Event
Rem Management action to log some event data to a
Rem file.
Rem The %EVENT..% variables contain values from the
Rem event that invoked the action.

echo LOG,%EVENT_YYYYMMDD%,%EVENT_TIME8%,%EVENT_NODEID%,\
    %EVENT_USERID%,%EVENT_TEXT% >>c:\temp\err.log
```

Message Enhancement

Event Management enhances messages by providing the origin of each message along with the message text. You can also customize the message text to meet the specific characteristics of your enterprise.

The following [message action definition](#) (see page 38) keywords let you control the message text that appears on the Event Console:

- EVALUATE
- FORWARD
- SENDKEEP
- SENDOPER
- WAITOPER

Correlation of Event Console Messages

Correlation is the ability to relate two or more previously unrelated events.

It is often true that a single message coming across the Event Console is not important unless seen in context with other messages. By constructing a series of message records and actions, you can be notified and take action if two or more events occur that together have more significance to your enterprise than any one event in isolation.

If you have CA NSM, you can use the CORRELATE action in the Event Console to take action on two or more messages to correlate them. Processing for this type of correlation can take advantage of messages issued from CA Common Services for z/OS components.

Example:

You have two UNIX computers in your accounting department. If one of these computers goes down, it is a problem and you probably have established Problem Management policies to deal with such an occurrence. However, should the second computer also go down, the problem becomes critical and the action you want to take in this situation may be quite different.

A solution to the problem would be to define message records that trap the event messages coming to the Event Console informing you that Accounting Machine #1 and Accounting Machine #2 are coming down. You can then define message actions for each message record that test for the occurrence of the other event message. You will now be automatically notified of the critical situation that exists in the Accounting department.

Sample procedures for two approaches to defining Event Management policy that correlates two incoming event messages are provided. See Enterprise Management in the online help.

Event Management Policy Activation

For performance reasons, the Event Management service reads all message records and message action definitions (the policy) from the Event Management database tables during startup, and maintains active lists of message records and message actions in memory resident tables. Database updates are not immediately reflected in these memory resident tables. Subsequent changes to message records and message actions do not take effect until the Event Management service refreshes the active version of the policy.

To refresh the active Event Management policy, issue the `opreload` command, or shut down and restart Event Management.

The `opreload` command directs Event Management to refresh the active message records and message actions immediately with the definitions stored in the Event Management database. Any requests for automated message processing remain in queue until the active policy is refreshed.

Note: For a step-by-step procedure for putting the Event Management policy into effect, see Enterprise Management in the online help.

Message Traffic Monitoring

Event Management gives you a visual window into message activity that lets you view and immediately respond to events as they occur, providing an unprecedented level of control over your systems. The Event Console can process events for any CA Technologies platform, UNIX, and Windows systems. Likewise, Event Consoles on other CA Technologies platforms can be used to display and manage z/OS events.

You access the Event Console by expanding the Enterprise Management tree and selecting Console Logs to open the Console Log window.

Note: For more information about monitoring message traffic, see Enterprise Management in the online help.

Console Log Window

The Console Log window provides two panes for messages.

Held Messages

The Held Messages pane displays messages that require a response. These messages are often critical and require immediate attention or require an operator reply. Held messages that require an operator reply are WTOR messages.

The Held Messages pane also displays messages that are sent using the SENDKEEP command. These messages can be removed by issuing a DELKEEP command.

Note: If a WTOR message has been sent, and either the Event Manager or the entire system goes down while the message is still pending, the message is queued and activated automatically (appears to be still active) when the Event Manager is brought back up.

Log Messages

The Log Messages pane displays all logged messages (including held messages).

Note: Through message records and message actions, these messages can be highlighted with a variety of user-specified colors and attributes to make them more noticeable on the display.

Event Console Log

The Event Console Log provides both a real-time and a historical online interface to all Event Management message activity. Created daily, the log is a file containing all messages written to the Event Console Log on a particular day. You can view the log for the previous day or the log for the next day (relative to the date of the log you are currently viewing), or select a log created on a specific date.

Only Event Management can read these files. You can set the directory where these files are stored with the Console Files Directory (CAI_CONLOG) environment variable.

Note: For more information about the CAI_CONLOG environment variable, see the *Reference Guide*.

You should consider a policy for archiving outdated Event Console Log files to tape for off-site storage.

Console Customization

Event Management provides a number of features that make viewing messages easy. These features let you move through the Event Console Log and narrow the focus of the display so that you can concentrate on the messages that are pertinent to your immediate situation.

Note: For the step-by-step procedures to customize your console, see the online help by drilling down through Enterprise Management, Event Management, Event Management Procedures, and Customize Your Console.

The following sections describe the main features that you can customize. Other customizable features, described in detail in the online help, include:

- Console log rescan interval
- Timer interval during which a blinking message is on or off

Command Buttons

You can define each of the sixteen numbered command buttons to execute a command when you click the button.

- When you click an auto-command button, the command is processed immediately and the corresponding text is written to the Event Console Log.
- When you click a manual command button, the command text appears on the command line. You can then edit the text before pressing Enter to issue the command. After you issue the command, the corresponding text is written to the Event Console Log.

Command History File

The command line of the Event Console stores the last 20 unique commands issued by the operator in a history file. To view the history file, click the drop-down list box arrow. You can select historical commands for processing by clicking the desired entry. The command you select appears on the command line where you can edit it, or press Enter to submit it for processing.

Record Selection

You can filter messages from the log by supplying criteria such as node name, user ID, source or workstation name, or text of a message to control which messages appear. Only those messages meeting all of the specified criteria appear on the console.

Store and Forward

The store and forward (SAF) facility stores messages locally when the Event Manager on a remote node is not available. When you enable SAF for a remote node and Event Management sends a message that cannot be delivered to the Event Console on that node, the message is added to a list and forwarded when the node is again available.

SAF is implemented by a daemon that periodically wakes up and tries to reconnect to all nodes for which there is an SAF list. The SAF list is maintained on a node and day basis in a directory specified by the `CA_OPR_SAF_ROOT` environment variable. Subdirectories are created for each node that is eligible for SAF and the stored messages are kept in them.

When an SAF message is finally sent to its destination, its message text is prefixed with the following:

```
QMSG date time
```

The prefixes to these messages let message actions differentiate between timely messages and those delivered after some delay (were stored).

After a message is successfully forwarded, it is deleted from the SAF list.

Note: For details about implementing SAF, see the post-installation tasks for Event Management in the *Installation Guide*.

Multiple Remote Console Logs

On a z/OS computer, local and remote consoles can be viewed through the GUI. The consoles for all nodes running Event Management and connected by CAICCI to the z/OS computer are available for viewing. You can restrict the nodes by adding a node list file to the `/cai/nsmem/emsvrc/data/` directory. You can use the `nodelist.sample` file in that directory as a model. Restricting the nodes improves performance.

Note: For more information about post-installation Event Management tasks, see the *Installation Guide*.

SNMP Facilities

SNMP is a widely used standard in network event management. It provides a method to identify objects in a network so that information concerning their status and activities can be monitored and reported on.

The daemons, commands, and utilities provided with Event Management give you an unprecedented level of control for locating, analyzing, troubleshooting, and reporting on the activity in your network.

SNMP Traps

An SNMP trap is, typically, an unsolicited message that reports on one of two types of events:

- Extraordinary events indicating that something is wrong or an error has occurred
- Confirmed events providing information about status, such as a process ending normally or a printer coming online

A variety of SNMP agents are available, including those provided through Agent Technology. They vary greatly in purpose, complexity, and implementation.

Despite their differences, all SNMP agents have the ability to:

- Respond to SNMP queries
- Issue an SNMP trap
- Accept instructions about where to route SNMP traps

Accepting instructions on where to route SNMP traps is typically referred to as accepting a setting for a trap destination. Setting the trap destination is important because traps should be directed to where they can be acted on.

Recognizing this, many vendors provide facilities for setting a system-wide default trap destination through an SNMP configuration file. For example, some UNIX platforms set their trap destination in the file `/etc/snmpd.conf`. This path and file name may be different for your system.

Accepting a trap destination setting, however, is only half the job. There also must be something at that trap destination that can receive and process that trap. Enterprise Management provides an agent called the CA trap daemon, `catrapd`, which can receive and process any traps directed to the destination (computer) where it is executing.

Any SNMP trap that `catrapd` receives is unpacked (decoded) and sent to the other Event Management components for processing. As part of this decoding, character representations, or strings, can be assigned to substitute names for the enterprise IDs that are part of the SNMP trap. CA Common Services for z/OS provides the following translation files for that purpose:

```
$CAIGLBL0000/snmp/dat/enterprise.dat
```

The file is self-documenting. You can add additional entries to this file by using a text editor.

Note: For more information about `catrapd` and SNMP usage, see the *Reference Guide*.

SNMP Traps Implementation

CA NSM has the ability to send and receive SNMP traps. Any received traps will appear on the Event Management console.

To receive traps, the daemon `catrapd` must be running. By default, the process listens on port 161. Your TCP/IP procedure has a PROFILE DD that contains reserved port numbers. If you are not currently using this port, locate and change the following line in your PROFILE's PORT section:

```
161 UDP SNMPQE
```

Insert a semi-colon at the beginning of the line as follows :

```
:161 UDP SNMPQE
```

If port 161 is unavailable, you must select the port you want and in file `/cai/nsmem/snmp/scripts/envset`, locate and change the following:

```
# port number to listen on  
CAICATD0001=161  
export CAICATD0001
```

Change 161 to 9161 as follows:

```
# port number to listen on  
CAICATD0001=9161  
export CAICATD0001
```

You can use the `catrap` program to send a trap. The syntax for the command is described in the *Reference Guide*.

catrap Command

The `catrap` command can issue SNMP traps to any destination in your network. It supports all the operands accepted as Open Systems standards for an SNMP trap command and can be used interactively, through z/OS USS, UNIX and Linux shell scripts or in Windows `.bat` files, or as part of an automated event handling policy defined to Event Management.

The operands provided as destination and information data to the `catrap` command convert automatically to the Open Systems standard datagram and are sent to the trap destination.

The `catrap` command lets user applications, shell scripts that are part of production jobs, or Event Management policies issue SNMP traps by executing the command and passing it the arguments.

Also, unlike some other SNMP trap commands, the `catrap` command does not restrict itself to any particular set of ISO or enterprise management information bases (MIBs), and is totally open for use with any MIB or pseudo MIB with no dependencies on any third-party network management components.

Note: For more information about `catrapd`, the `catrap` command, and SNMP usage, see the *Reference Guide*.

Management Information Base

A *management information base* (MIB) number identifies an event and includes other data to describe the object affected by the event.

Because the MIB number is such an important part of identifying an event, vendors must not use the same MIB number to describe different events.

To avoid the use of the same numeric codes by multiple vendors, standards exist that result in MIBs being organized into one of three broad categories.

- The first category of MIB is the *industry standard MIBs*. These are sanctioned and published by the International Standards Organization (ISO).
- The second category of MIB is the *enterprise MIBs*. Enterprise MIB numbers are assigned by the Internet Assigned Numbers Authority (IANA) to a given organization and are reserved for the exclusive use of that organization.
- The third broad category of MIB is the *pseudo-MIBs*. They are not sanctioned by or assigned by the IANA, but can be just as meaningful and useful as an ISO or enterprise MIB.

Pseudo-MIBs often piggy-back on the enterprise MIB of another organization and take advantage of many of the defaults available on a given platform.

Example:

Consider a pseudo-MIB that describes an *event tree*, in which each element on the tree represents information that could be sent when specified as a variable on the `catrap` command. At the top level, Enterprise 999 has two sub-trees, Databases (999.1) and Financial Applications (999.2). The General Ledger database and application are identified as 999.1.1 and 999.2.1 respectively, and they have branches that represent various related events.

999.1.1.1, .2, and .3

Represent General Ledger database shutdown, start, and journal full.

999.2.1.5, .6, and .7

Represent General Ledger application warm start, cold start, and error.

Sending a trap of 999.1.1.2 is equivalent to sending the message "The enterprise database server that handles the General Ledger database has been started." A trap of 999.1.1.3 indicates that the General Ledger database has encountered a journal full condition. And a trap of 999.2.1.5 indicates that the General Ledger application has resumed processing after a temporary outage (warm start).

Taking the example a step further, assume that CA Common Services for z/OS is executing on several nodes in this network, but you have decided that all SNMP trap traffic should be directed to a single monitoring computer, the Earth server. The Earth server receives the SNMP traps, which are recorded and acted on by the Event Management component.

Another computer in the network is used for the production financial applications. This is the Mars server. For some unknown reason, an error occurs and the General Ledger production application running on Mars terminates with an error.

Testing the return code issued by the General Ledger production executable, the shell script realizes that the exit code indicates a problem and issues an SNMP trap to alert the Earth server that something has gone wrong by executing the following command:

```
catrap earth "" "" 6 0 22 999.2.1.7 integer 128
```

catrap earth

Sends the identified trap information to the Earth server.

"" and ""

Instructs catrap to take the default Enterprise code and the default agent address respectively for this node.

6

Indicates that this command is sending a specific trap.

0

Identifies the specific trap number for this example.

22

Defines an arbitrary number we have selected as a timestamp indicator.

The next operands identify the variable binding (varbind) information for the trap.

999.2.1.7

Defines the ID of the object about which information is being sent. According to the event tree described earlier, this refers to an error in the enterprise financial application, General Ledger.

integer 128

Provides additional information about the event. It could mean “send an integer value of 128 to the Earth server,” assuming 128 is an error code that has meaning to the General Ledger application; or perhaps it is the exit code that the shell script detected as indicating an error.

When received at the trap target server Earth, the Event Management component (catrapd) decodes the event and performs automatic actions in response.

Returning to the event tree, you can see what other types of events could be sent, such as 999.1.1.1, indicating that the enterprise data server database for the General Ledger system has shut down.

When coupled with the other capabilities of CA Common Services for z/OS, the possibilities expand.

For example, you can use the Event Management facilities to intercept the error messages from any application and automatically execute customized catrap commands in response. The Workload Management function can detect key events and send traps in response to files becoming available for processing or applications completing their processing. Security violation attempts detected by Security Management can result in other SNMP traps being sent, and so on.

When on the receiving side of an SNMP trap, you can use Event Management message handling policies to:

- Open problem tickets automatically
- Send warning messages in human readable form to other consoles or terminals
- Start recovery jobs
- Post dependencies as having been met so that other production jobs can proceed
- Issue additional SNMP traps to one or more other nodes

As this simple example demonstrates, the possibilities for using SNMP trap information are virtually endless.

Note: For more information about the catrap command, including an example of how to issue an SNMP trap using the catrap command, see the *Reference Guide*.

Security

Before a command is executed from the console GUI or through a message action using oprcmd, an authorization check is performed to verify that the user ID is permitted to execute the command.

On z/OS, the identity of the user entering the command is deduced, and the command and all the resources it accesses are performed under the context of the original user. Generally, this is performed in conjunction with the security controls available with your system security software such as CA ACF2, CA Top Secret, or RACF.

On other platforms:

- If CA Technologies security is not active, the user ID issuing the command must be in the "Users authorized to issue commands" list before the command can be executed. You can edit this list in the Event Management tab of the EM Settings notebook.
- If CA Technologies security is active, CA Technologies security must explicitly permit the user ID to execute that command. To do this, you must create a rule for the CA-CONSOLE-COMMAND asset type.

OMVS Security

Certain programs that validate user identities must be identified by the authorized program facility (APF). Event Management installation scripts marked the appropriate programs using the "extattr +ap" command.

The user ID that Event Management runs under must have read access to the BPX.DAEMON, BPX.SERVER, and BPX.SUPERUSER resources and be assigned UID(0). The ID that runs the Java server and web server must have UID(0). Depending on your z/OS release level and the specific security options you have specified, you may need access to the BPX.SERVER resource, and the user ID that Event Management runs under may need surrogate permission to any user that is to be used with Event Management.

If your site requires accurate accounting information in order to use the Event Management processes, the OMVS segment for the user should contain a valid account code (for example, RACF:WORKATTR INFORMATION should contain WAACNT=xxxxx).

Command Submission on Behalf of Another User

Event Management lets you submit commands on behalf of another user. To enable this policy, you must supply new privileges for the user ID and for those users already logged on.

Note: For the step-by-step procedure, see the online help by drilling down through Enterprise Management, Event Management, Event Management Procedures, and Submitting a Process as Another User.

Protecting Event Management under HFS Security

For sites implementing HFS security using CA Top Secret or CA ACF2, you might need to write rules to protect the resources. This section describes the resources that the various started tasks and the end users require to use Event Management.

Web Server Requirements

Web Server requires read access to:

- IBMFAC BPX.CAHSF.SET.RLIMIT and BPX.CAHFS.SET.PRIORITY
- HFSSEC /CAI.NSMEM.BROWSER
- HFSSEC /CAI.NSMEM.BROWSER.IMAGES
- HFSSEC /CAI.NSMEM.BROWSER.CLASSES
- HFSSEC /CAI/NSMEM.BROWSER.SCRIPTS
- HFSSEC /CAI.NSMEM.WV.HTML
- HFSSEC /CAI.NSMEM.RW.CONFIG
- HFSSEC /CAI.NSMEM.WV.SCRIPTS

Web Server requires update/alter access to:

- HFSSEC /CAI.NSMEM.RW.TMP

NSMJSERV (Java Server) Requirements

NSMJSERV (Java Server) requires read access to:

- HFSSEC /CAI.NSMEM.ATECH.SCRIPTS
- HFSSEC /CAI.NSMEM.BROWSER.SCRIPTS
- HFSSEC /CAI.NSMEM.CAL.SCRIPTS
- HFSSEC /CAI.NSMEM.EMSRVC.SCRIPTS
- HFSSEC /CAI.NSMEM.HELP.SCRIPTS
- HFSSEC /CAI.NSMEM.OPR.SCRIPTS
- HFSSEC /CAI.NSMEM.ROUTER.SCRIPTS
- HFSSEC /CAI.NSMEM.SECU.SCRIPTS
- HFSSEC /CAI.NSMEM.SNMP.SCRIPTS
- HFSSEC /CAI.NSMEM.STAR.SCRIPTS
- HFSSEC /CAI.NSMEM.COMMON.SRC
- HFSSEC /CAI.NSMEM.MESSAGES

NSMJSERV (Java server) requires exec access to:

- HFSSEC /CAI.NSMEM.BIN
- HFSSEC /CAI.NSMEM.BIN
- HFSSEC /CAI.NSMEM.LIB
- HFSSEC /CAI.NSMEM.EMSRVC.BIN
- HFSSEC /CAI.NSMEM.WV.SCRIPTS
- HFSSEC /CAI.NSMEM.WV.BIN

NSMJSERV (Java server) requires update/alter access to:

- HFSSEC /CAI.NSMEM.RW.TMP
- HFSSEC /CAI.NSMEM.RW.CONFIG

NSMEMSTR Requirements

NSMEMSTR requires read access to:

- HFSSEC /CAI.NSMEM.SCRIPTS
- HFSSEC /CAI.NSMEM.ATECH.SCRIPTS
- HFSSEC /CAI.NSMEM.BROWSER.SCRIPTS
- HFSSEC /CAI.NSMEM.EMSRVC.SCRIPTS
- HFSSEC /CAI.NSMEM.HELP.SCRIPTS
- HFSSEC /CAI.NSMEM.OPR.SCRIPTS
- HFSSEC /CAI.NSMEM.ROUTER.SCRIPTS
- HFSSEC /CAI.NSMEM.SECU.SCRIPTS
- HFSSEC /CAI.NSMEM.SNMP.SCRIPTS
- HFSSEC /CAI.NSMEM.STAR.SCRIPTS
- HFSSEC /CAI.NSMEM.COMMON.SRC
- HFSSEC /CAI.NSMEM.CAL.SCRIPTS
- HFSSEC /CAI.NSMEM.MESSAGES

NSMEMSTR requires update/alter access to:

- HFSSEC /CAI.NSMEM.RW.TMP
- HFSSEC /CAI.NSMEM.RW.CONFIG
- HFSSEC /CAI.NSMEM.RW.LOGS

NSMEMSTR requires exec access to:

- HFSSEC /CAI.NSMEM.BIN
- HFSSEC /CAI.NSMEM.LIB
- HFSSEC /CAI.NSMEM.STAR.BIN

Java GUI Requirements

Use of the Java GUI requires read access to:

- HFSSEC /CAI.NSMEM.RW.CONFIG
- HFSSEC /CAI.NSMEM.RW.LOGS
- HFSSEC /CAI.NSMEM.RW.DATA
- HFSSEC /CAI.NSMEM.SECOPTS

Use of the Java GUI requires exec access to:

- HFSSEC /CAI.NSMEM.BIN
- HFSSEC /CAI.NSMEM.LIB

Use of the Java GUI requires update/alter access to:

- HFSSEC /CAI.NSMEM.RW.CONFIG.DEBUG\$LOG

Event Management Utilities

The CA Common Services for z/OS Event Management Utilities provide services that enable mainframe applications to integrate with Event Management running on z/OS and other platforms, through z/OS assembler and C language interfaces.

Frequently, applications running on z/OS need to route alerts to Event Management. Often, these applications run in environments where directly generating Event Management alerts is difficult or impossible. Because there are many ways to integrate with Event Management, it can be difficult for any particular application to support all possible configurations. The Event Management Utilities provide a way of achieving CA Technologies integration in most of these cases.

Tasks running in most z/OS environments—such as TSO, batch, or started task address spaces, as well as IDMS, CICS, and IMS transactions—can invoke Event Management Utilities. No special privileges are needed, and assembler programs can use the utilities without requirements for IBM's Language Environment (LE) or USS. To guarantee top performance, most processing occurs asynchronously under the direction of a separate utility server process in its own address space. This design results in a service that can be used by nearly any application, including those with strict performance constraints.

The overall utility interface consists of three components:

- A macro interface (#EMWTO) that can be invoked by any mainframe application
- A C/C++ interface library for applications written in C and other LE-conforming languages
- An Event Management Utility process that routes requests to Event Management on any CA Technologies platform

How the Event Management Utilities Process Works

When an application generates an Event Management message, it uses one of the supplied application program interfaces (APIs) (assembler or C/C++) to direct messages to the Event Management Utilities. Depending on the particular platforms involved, there can be many different techniques for communicating with Event Management. Currently, these Event Management transports are supported:

SNMP

Enables messages to form formal SNMP traps, which can then be routed to Event Management or other network managers.

SYSLOGD

Enables messages to be routed to the syslog daemon. The daemon is an IBM-supplied UNIX process that can be configured to route messages to Event Management.

CA Common Services for z/OS Event Management

Enables messages to be processed directly without using a transport such as SYSLOGD, WTO, or SNMP. Event Management has an optional store-and-forward feature that guarantees message delivery.

The Event Management Utilities evaluate each message against a user-supplied configuration policy, and depending on the options selected, reformats and broadcasts the message to Event Management using one or more of the message transports. This flexibility provides you with granular control over message processing and integration with Event Management.

Use the Event Management Utilities configuration statements to:

- Selectively route messages to Event Management according to pattern matching on the message text and other message context fields.
- Suppress or ignore certain messages.
- Reformat messages, for example, by converting between different code page values.
- Selectively redirect messages to a variety of non-CA Technologies systems using one of the message transports.

Note: Wherever possible, applications should interface directly with Event Management rather than using Framework Event Management Utilities, which is intended for special cases when an application cannot use Event Management directly.

Prerequisites

The following minimum prerequisites apply:

- z/OS version 1.6
- IBM LE, Framework Event Management Server, and C/C++ applications
- IBM TCP/IP (for SNMP or direct Event Management interfaces)
- CAICCI and its prerequisites

Configuration Statements

The Event Management Utilities enable you to have many instances of Event Management running across the enterprise and to associate messages with each of these instances on a selective basis. Depending on your configuration, messages can be routed to different instances of Event Management according to message content or attributes such as language. Two types of statements define how messages are associated with Event Management destinations:

Criteria statement

Tests incoming messages and associate them with different Event Management destinations.

Destination statement

Identifies which instance of Event Management is to receive a particular set of messages and the technique that should be used for communications.

You can store criteria and destination statements in data sets or in HFS or zFS files.

If you use data sets, you can identify them in the Event Management Utilities CAW0JCL member CAS9FEMS as follows:

```
//CRIT DD DSN=criteria_dsn,DISP=SHR  
//DEST DD DSN=destination_dsn,DISP=SHR
```

If you use HFS or zFS files, you can identify them in the Event Management Utilities CAS9FEMS member as follows:

```
//CRIT DD PATH=criteria_file_name  
//DEST DD PATH=destination_file_name
```

If you start the Event Management Utilities as a UNIX process, you use the `cas9fems` parameters to identify the HFS or zFS files.

The syntax for criteria and destination statements is provided in the sections that follow.

Destination Statements

Each of the logical destinations specified in your criteria statements must be defined in the destination statement file. The statements in this file define the logical destinations and optionally explain how Event Management Utilities should exchange messages when communicating with this destination.

A destination statement has the following format:

```
destination_name: driver(driver) [lang(language)] [drvdata(data1)]  
[drvropts(data2)] [mlwto]
```

destination_name

Defines a logical identifier for use by the criteria control statement file.

Limits: 64 characters

driver

Specifies the name of a program that is to be the driver for this destination. Different [Event Management Utilities drivers](#) (see page 68) are available to communicate with Event Management using a variety of communication techniques.

language

(Optional) Specifies the character-encoding scheme of the message. It represents a locale or code page name acceptable to the IBM iconv() function.

Default: Messages are routed to the destination unchanged.

Limits: 16 characters

In some cases, translation is not possible between the specified code page and the code page that Event Management uses. This can occur, for example, if you specify multi-byte messages (such as UCS-2 or IBM-939) and Event Management runs with a code page that lacks translation for certain input values.

data1

(Optional) Passes information to the driver program. For example, the SNMP driver uses this parameter to let you specify the object identifier (OID) used in generating SNMP traps.

data2

(Optional) Passes information to the driver program. The Event Management driver allows you to pass a remote nodename in the format (-n XYZNODE) to forward a message directly to another node without having to define a message action. The SNMP driver allows you to pass a port number in the format (-p nnnn) to specify the port where the trap will be sent.

mlwto

(Optional) Enables the processing of multi-line WTO messages. This parameter is mlwto (case-sensitive).

Criteria Statements

If you have multiple instances of Event Management running on different computers in your enterprise, you can use criteria statements to filter messages according to attributes or content. Messages that satisfy a criteria statement are then routed to a logical Event Management destination that you specify using destination statements.

A criteria statement has the following format:

```
destination_name: text(text_pattern) sev(severity) [lang(language)]
```

destination_name

Specifies a logical identifier that matches a name configured in the destination statement file.

text_pattern

Specifies a pattern that message text must match in order to be routed to the destination. Full support is provided for wildcards using UNIX style regular expression syntax, permitting flexible message selection according to a wide range of criteria. If you do not specify a pattern, all messages are routed to the destination according to severity and language restrictions.

Note: By default, the Event Management Utilities support the syntax commonly referred to as “basic regular expression.” If you want to use the “extended regular expression” syntax, then you must use the `-e` start parameter to run the Event Management Utilities.

severity

(Required) Specifies the severity of messages to be routed to this destination.

Limits: INFO, WARN (warning), CRIT (critical error), IMM (immediate action), or asterisk (*) (any severity)

language

(Optional) Specifies the character-encoding scheme of the message. It represents a locale or code page name acceptable to the IBM `iconv()` function.

Default: Messages of all encodings are eligible for the destination.

Limits: 16 characters

Example: Configuration Statements

These following two files set up the criteria and destinations for the messages:

CRITERIA.DAT

Defines the criteria for messages sent to each of 5 destinations. All warning messages are routed to the first destination. All informational messages with the prefix CAKO are sent to the second destination. The third destination receives German language messages with the prefix CAS9nxxE where n is a number between 6 and 8 and xx represents any two characters. Destination 4 receives all messages with the prefix IEF40 except those with the prefix IEF403. All messages that do **not** have the prefix XCOM are routed to destination 5.

```
DestId1: text(".") sev(WARN)
DestId2: text("^CAKO") sev(INFO)
DestId3: text("^CAS9[6-8].E") sev(*) lang(IBM-273)
DestId4: text("^IEF40[^3]") sev(*)
DestId5: text("(^XC0[^M])|(^XC[^0])|(^X[^C])|(^[^X])") sev(*)
```

DEST.DAT

Defines the five destinations referenced above. Messages to the fourth destination are translated from the IBM-273 code page (German language) to the IBM-1047 code page (English). The second, third, and fifth destinations are remote nodes.

```
DestID1: driver(/cai/nsmem/bin/casyfem1)
DESTID2: driver(/cai/nsmem/bin/casyfem1) drvropts(-n ABCNODE)
DestID3: driver(/cai/nsmem/bin/casyfem1) drvropts(-n XYZNODE)
DestID4: driver(/cai/nsmem/bin/casyfem4) lang(IBM-1047)
DestID5: driver(/cai/nsmem/bin/casyfem5) +
  drvropts(-p nnnn) +
  drvrddata(1.3.6.1.4.1.791.2.9.2.2 6 12 1.3.6.1.4.1.791.2.2.3.1)
```

Where /cai/nsmem/bin depends on the installation location.

Event Management Utilities Drivers

Depending on your configuration, different driver programs are available to enable integration with Event Management. The following drivers are available:

CASYFEM1

Invokes Event Management directly on z/OS. It routes events directly to Event Management on the local system.

If you use the optional parameter `drvropts(data2)` you can forward a message directly to a remote node specified by `data2` without defining a message action. For example:

```
CON11: driver(/cai/nsmem/bin/casyfem1) drvropts(-n XYZNODE) [mlwto]
```

If you use the optional parameter `mlwto`, you enable the processing of all lines of multi-line WTO messages. For more details, see [Processing Multi-line WTO Messages](#) (see page 71).

CASYFEM4

Integrates with Event Management using UNIX style syslogd calls. `syslogd` must be configured to route messages to Event Management, either locally or on another platform.

CASYFEM5

Generates SNMP traps that can be processed on other CA Technologies platforms through integration with Event Management. To specify the OID that the SNMP trap will produce, you must code the OID value using the `drvdata` parameter in the destination statement. A listening port number may be specified in the `drvropts` parameter in the destination statement. The default port is 162. For more information, see the *catrap* discussion in the Event Management chapter of the *Reference Guide*.

Example

The following statement creates a destination called `SNMP` with a specific OID value:

```
SNMP: driver(/cai/nsmem/bin/casyfem5) +  
      drvdata(1.3.6.1.4.1.791.2.9.2.2 6 12 1.3.6.1.4.1.791.2.2.3.1)
```

The sample `drvdata` value is as follows:

- 1.3.6.1.4.1.791.2.9.2.2 is the enterprise top-level OID
- 6 specifies the generic trap type
- 12 specifies the enterprise-specific trap subtype
- 1.3.6.1.4.1.791.2.2.3.1 is the OID

The trap will be sent to port 162 since the parameter `drvropts` was not used to specify a different port.

Run the Utilities as an OMVS Process

You can configure Event Management Utilities to run as an OMVS process using job control language (JCL). A sample is provided in the CAWOJCL(CAS9FEMS) data set member.

To run the server as an OMVS Process

1. Make a copy of the CAWOJCL(CAS9FEMS) member.
2. Edit the copy to suit your requirements. If Event Management store-and-forward is to be implemented, edit the job stream according to the instructions in the sample.

The following parameters can be used:

-b

Specifies basic regular expression testing. This is the default.

-e

Specifies that extended regular expressions are used in criteria statements (mutually exclusive with -b).

-v

Requests detailed diagnostic trace messages.

-t

Runs a test.

The server job is created to execute the CAWOPLD(CAS9FEMS) program.

3. Submit the server job.

The server starts.

To stop the server, issue the STOP command:

```
/P TESTFEMS
```

Route z/OS Console Messages to Event Management Utilities

You may wish to obtain messages from the system console to be processed under Event Management or CA NSM. Event Management Utilities (CAS9FEMS) uses a method that incorporates an MPF exit on the mainframe to trap the messages and forward them to the Event Management Utilities.

Once the message is in FEMS, you have the option of how and where to treat the message.

To trap messages on the mainframe and forward them to FEMS

1. Test the FEMS server.
 - a. Ensure that CAICCI is active on the system.
 - b. Create the DEST and CRIT DD files as documented in the *Administration Guide*.
 - c. Update CAWOJCL member CAS9FEMS to match your installation standards.
 - d. Use the parms `-t -v` to run a test in verbose mode.
2. Start the FEMS server. After the server has been tested successfully it should be converted into an STC. Remember to use a user ID that has a valid OMVS segment in the security record. If using Event Management store-and-forward in conjunction with CAS9FEMS, the user ID should belong to the same OMVS security group as the ID that was used to install Event Management.

3. Activate the MPF exit. Update the MPFLSTxx member to include the desired messages. The message should include the USEREXIT(CATNMPEX) operand to call the exit to process the message. For example, to trap message IEF450I you would code:

```
"IEF450I,SUP(NO),USEREXIT(CATNMPEX)"
```

4. Update the CRIT and DEST files to handle the additional messages. For example, to capture an IEF450 message, add the following to the CRIT file and a corresponding line:

```
CON11: text("^IEF450") lang(*) sev(*)
```

Add one of the following to the DEST file to send it to Event Management.

- To reach the local node:

```
CON11: driver(/cai/nsmem/bin/casyfem1)
```

- To reach a remote node XYZNODE:

```
CON11: driver(/cai/nsmem/bin/casyfem1) drvropts(-n XYZNODE)
```

5. Test the exit. Create the messages on the console and verify that they reach the desired destination.

When testing is complete, you must issue a STOP command to terminate the CAS9FEMS job. From the MVS console, issue /P TESTJOB where TESTJOB is the jobname.

6. Write message actions. This step may be required if the message is reaching Event Management.

If messages reach Event Management on the local node, you may direct them to remote Event Management nodes by defining Message Actions. Alternatively, messages may be sent directly to other Event Management nodes by using the drvropts parameter in the DEST.DAT file.

This process can be used to open a help desk issue based on console messages. Once this process is working to your satisfaction, you can forward the desired messages to Event Management, where a CA NSM Advanced Help Desk (AHD) issue can be opened. For more information on opening issues, see the *Unicenter TNG Administration Guide*.

Processing Multi-line WTO Messages

Each line of a multi-line WTO will appear on the Event Management console of the target node if the mlwto optional parameter is specified in the DEST.DAT file for the driver. The lines (major and minor) may be interrupted by other messages on the console. The Category field in the Message Record is used to tie the lines together.

If these messages require processing on the target node, the contents of the Category field provides the information needed to associate the lines of the multi-line WTO.

The Category field contains a WTO Descriptor built by CAS9FEMS. This field indicates whether the line is first, middle, or last. It also contains a version number and the identifier supplied with the WTO in the mainframe syslog to link together all the lines of the multi-line WTO.

The format of the 12-byte WTO descriptor is as follows:

WIIIIIIIXY

VV

Indicates the two-byte version '01'.

IIIIII

Indicates the eight-byte decimal character representation of the 3-byte binary message ID given to the MPF exit CATNMPEX by z/OS. The last 3 characters are displayed in the mainframe log to tie the lines of the WTO together.

X

Indicates the line-order, one of the following characters to indicate the order of the line:

- O - only one line (single-line WTO)
- F - the first line of a MLWTO
- M - the middle line of a MLWTO
- L - the last line of a MLWTO

Y

Indicates the line-type, one of the following characters to indicate what type of line:

- D - for data
- C - for control
- L - for label
- E - for end
- ? - for other

For more information, see the IBM documentation on the WTO MACRO for line type and line type referenced above.

Issue Console Commands Using CAconsole

You can use CAconsole to issue console commands from within the UNIX environment. This allows a message action to issue any required console commands directly without having to get automation into the process. The basic syntax of the command is as follows:

```
CAconsole [-attribute]
```

-v

Indicates to return any output.

-t

Indicates the amount of time to wait for response.

-i

Indicates the text of the command.

-x

Indicates to translate the command to upper case.

For example:

```
CAconsole -t5 -v -i"D OMVS,0"
```

In the following example, first determine if any job in a set of jobs is running. If any of the jobs is running, send an alert to a remote NSM machine and then cancel the job.

```
cd /cai/nsmem
# Set environment variables
. PROFILE
# Issue a Display Job,list command and place the output into a file display.out
CAconsole -I"D J,L" -V >display.out
# Search the output file for any job PRRP913x and place any hits into display1.out
grep PRRP913 display.out > display1.out
# If none are found send a message.
if [ $? = 1 ];then
  cawto -n NSMI Job PRRP913 is not present
else
  cawto -n NSMI Job is present, will cancel
# We need to parse the output to capture the job information.
  cat display1.out | while read v1 v2 v3 v4 v5 v6 v7 v8 v9 vx ;
  do
    echo "$v1\n $v2\n $v3\n $v4\n $v5\n $v6\n $v7\n $v8\n $v9\n" >display2.out
    job=`grep PRRP913 display2.out`
    echo $job
    job=`grep PRRP913 display2.out`
    echo $job
    CAconsole -I"CANCEL $job"
  done
fi
```


Chapter 3: Agent Technology

This chapter provides a general explanation of agents and their operation and describes the architecture of CA Common Services for z/OS Agent Technology.

Note: For more information about Agent Technology operation and configuration, see the CA NSM documentation.

This section contains the following topics:

[Overview](#) (see page 75)

[How an Agent Works](#) (see page 76)

[Components](#) (see page 77)

[Installation Considerations](#) (see page 79)

[Start Agent Technology](#) (see page 81)

[Mainframe Agent Installation Considerations](#) (see page 82)

[Performance Considerations](#) (see page 86)

[Configuration](#) (see page 89)

[Turn off Statistics Collection](#) (see page 90)

[Integration with CA OPS/MVS Event Management and Automation](#) (see page 91)

Overview

Agent Technology enables you to achieve enterprise-wide, automated, high-level monitoring and management of critical resources including hardware, software applications, and network devices. This technology consists of agents that gather data and take action on behalf of managers; powerful managers that use this information to initiate and control activity and delegate authority; and an infrastructure of common services that enable the flow of information between them.

CA Common Services for z/OS provides the essential infrastructure that makes possible the operation of Agent Technology in the z/OS environment. This infrastructure supports pre-packaged z/OS agents such as the z/OS system agent, MQSeries agent, CICS agent, CA-IDMS agent, and DB2 agent, and z/OS managers such as CA Automation Point, and CA NSM System Status Manager CA OPS/MVS Event Management and Automation Option. Agents running on z/OS can be managed by a Distributed State Machine (DSM) running on a UNIX/Linux or Windows system.

Agent Technology supports a wide range of platforms and is deployable in client/server, internet, and intranet environments. This enables enterprise-wide monitoring and management of z/OS elements, and enhances the ability of z/OS environments to participate in a true heterogeneous network.

How an Agent Works

Many devices—such as mainframes, personal computers (PCs), printers, routers, hubs, and interfaces—typically have built-in SNMP agents that monitor and gather information about the status of these resources so they can be managed.

An agent responds to requests issued by management applications or managers regarding specific data associated with a resource. These requests may involve, for example, information about file systems, databases, and memory usage. An agent also sends traps to managers to notify them of noteworthy conditions related to a resource. The managers interpret the raw data and take action based upon how each manager is configured.

Agent Protocol

A protocol enables information to be understood by managers and agents. CA Common Services for z/OS SNMP agents use the following protocols:

Communications protocol

Uses the User Datagram Protocol (UDP) of the Transmission Control Protocol/Internet Protocol (TCP/IP) suite.

Network management protocol

Uses the SNMP that runs on top of TCP/IP.

Both agent and management applications can view the collection of data items for the managed resource. This collection is known as the MIB. Each MIB has attributes that represent aspects of the managed resource. The MIB is the central point at which manager and agent processes converge; it is populated at run-time by configuration files and agent activity.

Managed Objects

Each resource that an agent monitors is called a *managed object*. A managed object can represent a physical device, such as a printer or a router, or it can represent an abstraction, such as the combination of hardware and software components that constitute a network connection between two nodes. A managed object can be monitored and, in most cases, controlled with the use of one or more management applications.

CA Common Services for z/OS groups managed objects into *classes*. A class is a group of managed objects that share a common definition, and therefore share common structure and behavior. By changing the behavior of a class, you can change the behavior of the managed objects that belong to that class.

Each Agent Technology agent has an agent .dat file. A .dat file contains the definitions for the managed object classes based on the monitored resources. These class definitions control the type of data that the agent monitors and collects.

Note: For information about the .dat files, their syntax, and modifications you can make, see the CA NSM documentation.

Agent Status

Every managed object has a state. A state is one of a set of predefined possibilities for the condition of the managed object (for example, up, down, or unknown). Agents collect data about their managed objects and send this data to the managers. Based on the rules defined in the agent configuration set, the agent processes the data and determines the state of each managed object.

An agent can signal a state change either by sending traps to an interested manager or as a result of a manager poll.

If you have CA NSM, a state change appears on the 2D Map of your Real World Interface as a change in icon color. You can drill down through the network, to the computer, to Unispace, to the agent type (CICS, MQSeries, and so on), and to the specific agent instance that experienced the state change. You can right-click the agent icon and select View Agent to view the detailed status information.

Components

The Agent Technology infrastructure provided by CA Common Services for z/OS supports the processes and communications between agents, objects, and managers. It consists of the following components:

- Service Control Manager (awsservices)
- Distributed Services Bus (aws_orb)
- SNMP Administrator (aws_admin)

Service Control Manager

The Service Control Manager starts and stops, in the correct order, all other Agent Technology components and agents on the node. While this component is running, you can install, start, stop, or uninstall the other components.

Services

Agent Technology for z/OS is comprised of the following services, which are a subset of the CA Technologies services on the distributed platforms:

awsservices

Service controller.

aws_orb

Object request broker (that is, the communications handler).

aws_sadmin

SNMP handler. This service is also responsible for maintaining the MIBs within the Object Store in the UNIX System Services (USS) zSeries File System (zFS).

All of the z/OS-based agents communicate with the Agent Technology for z/OS services, which in turn communicate with the various NSM machines. This communication occurs using the TCP/IP protocol. By default, the following ports are used by these services:

- 6665-UDP-type port used by sadmin for SNMP requests.
- 7770-TCP-type port used for ORB->ORB communications (Release 2.0-2.1).
- 7774-TCP-type port used for ORB->ORB communications (Release 2.2-3.0 and 11.0).
- 9990-TCP-type port used by awsservices (request port).
- 9991-TCP-type port used by awsservices (control port).

Review the TCPIP.ETC.SERVICES file to determine if any other service has been registered to use any of these ports.

You can also issue the netstat command (at the ready prompt in TSO) to determine if any of the default ports are already in use prior to starting the Agent Technology services.

Distributed Services Bus

The Distributed Services Bus serves as an object request broker for all the other Agent Technology components. All information exchanged between Agent Technology components is placed on and retrieved from the Distributed Services Bus.

SNMP Administrator

The SNMP Administrator is responsible for checking the community string and Internet Protocol (IP) address of get, get-next, and set requests to make sure they come from authenticated management applications. The `aws_sadmin` service encodes trap and response protocol data units (PDUs) from messages received from the `aws_snmp` listener process and sends them to management applications.

Installation Considerations

Note the following general installation considerations for Agent Technology for z/OS:

- Be sure to read all documentation.
- Pay special attention to the post installation tasks.
- Remember that UNIX is case-sensitive. Make sure all OMVS commands, operands, file-names, and so on are entered in the proper case.

Additional considerations are described in the following sections.

AWADMIN Account

One of the key steps to the successful installation of Agent Technology is the establishment of the administrator account (default name, AWADMIN) in the default group AWGROUP. The zFS files are associated with AWGROUP when they are formatted. The user account must be authorized to:

- Perform job submission
- Write to the HFS or zFS (OMVS access)
- Submit APPC transactions

This account must also specify a HOME directory equal to that of the root Agent Technology directory (`/cai/agent` by default). It is not necessary to use the exact Group and User codes (911 and 912 respectively) as those specified in the installation process.

Important! Do not use UID(0) for the administrator account.

Additionally, if you use the CA Top Secret product, you may want to amend the example TSS commands to add syntax to create a department specific to Agent Technology. For example:

```
tss cre(awdept) type(dept) name('Agent Technology Department')
```

This department name should then be referenced in the AWGROUP and AWADMIN definitions.

Security Requirements

Questions occasionally arise regarding the need to grant superuser privileges to user IDs that are used to run one or more of the z/OS-based agents. It is neither required, nor desired, for these IDs to hold this privilege. What is required however, is that these accounts must participate in the Agent Technology security group (default name, AWGROUP), as established in the Agent Technology installation process. Note that the zFS files are owned by the group AWGROUP once they are formatted during the install process. In addition, if you intend to run your agent as a started task, the Agent Technology security group must be the default group for the task's user ID.

Configure the aws_sadmin Service

The aws_sadmin service is responsible for sending SNMP traps to the distributed NSM machines. To achieve this goal, the aws_sadmin service must know the DNS name or IP address (and listener port) for the NSM machines. This information is stored within file aws_sadmin.cfg in directory \$AGENTWORKS_DIR/services/config/aws_sadmin. You must update this file with the appropriate DNS names or IP addresses of your NSM machines. Be sure to preserve the non-displayable tab characters (x'05) contained within this file. These characters are used as field delimiters on the SNMP_TRAP lines.

The aws_sadmin.cfg file is also used to define the SNMP community strings for the aws_sadmin SNMP service. By default, these strings are set to public (for read access) and admin (for write access). If your site requires more secure SNMP community strings, these names can be changed to meet your needs. Again, be sure to preserve the non-displayable tab characters (x'05) contained within this file. These characters are also used as field delimiters on the SNMP_COMMUNITY lines. Also note that any change to the SNMP community strings must be reflected in the configuration of the NSM machines. Be sure to make your NSM administrator aware of these changes. Note that an agent can have its own community strings specified in a configuration set, which is unique to that agent and which overrides the default community strings specified in the aws_sadmin.cfg file.

Environment File (ENVFILE)

The Common Services CAWOOPTV library contains a member named ENVFILE. This member is used by various Agent Technology utilities (as well as all of the z/OS-based agents) to set environment variables required to communicate with the Agent Technology services. The configuration data within this member is created when the Agent Technology services are installed, and must be accurate. This data is specific to the current LPAR, TCP/IP stack, and Agent Technology zFS. Any change to these items must be reflected in the ENVFILE member. In general, the values must match the equivalent values specified in the agentworks.profile script on the zFS.

Important! The Common Services CAWOOPTV library has special DCB attributes that must be maintained. These attributes are particularly necessary when reading the ENVFILE member. Communications problems will occur if the ENVFILE member is moved and read out of a standard fixed-block PDS.

Start Agent Technology

Agent Technology does not depend on any other CA Common Service. Once installed, it can be started any time after UNIX System Services (USS) and TCP/IP have started. None of the z/OS-based agents can be started until Agent Technology for z/OS has started, and all of its services have properly initialized.

You can use the Common Services CAWOPROC library member AWSTART to start the Agent Technology services or you can use online commands (shell scripts) to perform the same tasks. AWSTART can be run as a batch job or started task. You should run it as a started task in production.

Note: It usually takes 1-2 minutes (after the AWSTART job or task ends) for the Agent Technology services to initialize.

Note: Starting the Agent Technology Services under z/OS only starts the services and not the agents themselves. Each of the agent instances must be individually started and stopped in z/OS. This action usually occurs within the application being monitored, or through an MVS console command.

Operation

Some of the more useful batch jobs and PROCs let you start, stop, and control Agent Technology for z/OS, which are:

AWSTART

Starts Agent Technology for z/OS.

AWSTOP

Stops Agent Technology for z/OS.

AWSTATUS

Provides status information.

Note: Each batch job or PROC runs a shell script that resides on the Agent Technology zFS and the output for each job is written to the Agent Technology zFS.

Online Commands

To use online commands, enter the OSHELL through the TSO OMVS command and enter commands described in the *Reference Guide*.

Example: Command Output

The following output shows a status, start, a second status command and a list command. Assuming these are the first commands we enter after logon, the agentworks.profile is first executed to enable our current environment.

```
AWADMIN:/agent:> . agentworks.profile
AWADMIN:/agent:> awservices status -1 : AWSERVICES is not running
AWADMIN:/agent:> awservices start awservices started 704643086
AWADMIN:/agent:> awservices status
AWADMIN:/agent:> "awservices" is running
AWADMIN:/agent:> awservices list
AWADMIN:/agent:> RUNNING aws_orb:aws_orb RUNNING aws_admin:aws_admin
```

Mainframe Agent Installation Considerations

This section describes the mainframe agent installation considerations.

Agent MIB Files

Agent Technology for z/OS no longer supplies the MIB files for most of the z/OS-based agents. Each agent installation provides the MIB file for the agent installed. For ease of use, we recommend that you copy these files into the Common Services MIBLIB when the agent is installed by using the AWADDMIB sample in Common Services CAWOJCL library. The AWADDMIB sample adds the new agent MIB into the SMP/E environment as a ++USERMOD, which allows you to use the standard Agent Technology 'install_mibs' script whenever the Agent Technology Object Store is cleared and rebuilt.

If you want to work outside of SMP/E to install the agent's MIB, then edit the \$AGENTWORKS_DIR/services/tools/install_mibs script to point to the agent's own MIBLIB as the source of the MIB to load into the aws_admin store.

Communicate Between Agents and Services

All z/OS-based agents must communicate with the Agent Technology for z/OS services running under USS. The agents communicate to Agent Technology Services through a TCP/IP socket connection. To facilitate this communication, the following modifications must be made to the JCL job stream for the system being monitored:

- The Common Services load library must be added to the STEPLIB concatenation.
- The following DD statements must be added:

ENVFILE

Defines the pointer to the ENVFILE member within the CAW0OPTV library. This member contains configuration data that points the Agent to the appropriate TCP/IP Port and Stack and to the Agent Technology zFS.

You can copy and rename this member so that the member is unique to the system and to the TCP/IP stack that Agent Technology runs on.

SYSTCPD

Defines the pointer to the TCPDATA file for the local TCP/IP Stack. This is the same file referenced in your TCP Startup proc. This file contains configuration data about the TCP/IP stack.

- Add any log or configuration files mandated by the specific agent being installed.

The Discovery Process

You may encounter some problems with the discovery of agents on a z/OS LPAR. The following list provides a few tips for diagnosing and resolving these problems:

- Make sure your z/OS node is properly classified.

The first-level discovery process uses your TCP/IP stack's SNMP process to classify your mainframe node. This is not the Agent Technology SNMP service, but rather, this is the SNMP service that comes with the IBM (or third-party) TCP/IP stack. This service must be running at the time the mainframe node is discovered for the node to be properly classified as an IBM3090.

If the node is not properly classified, it can be reclassified manually from the NSM machine (using the reclass command).

- Community strings must agree between the z/OS `aws_sadmin` service and the NSM machine.

See [Configuring the `aws_sadmin` Service](#) in the Installation Considerations section previously in this chapter for more information.

- Is the z/OS SNMP service listener port bound?

Run `onetstat` under `OMVS` to ensure the SNMP listener port (6665 by default) is in the proper state. Check to make sure there are no firewalls preventing UDP traffic from entering your mainframe network.

- Is the `awsAdmin` MIB loaded in the z/OS store?

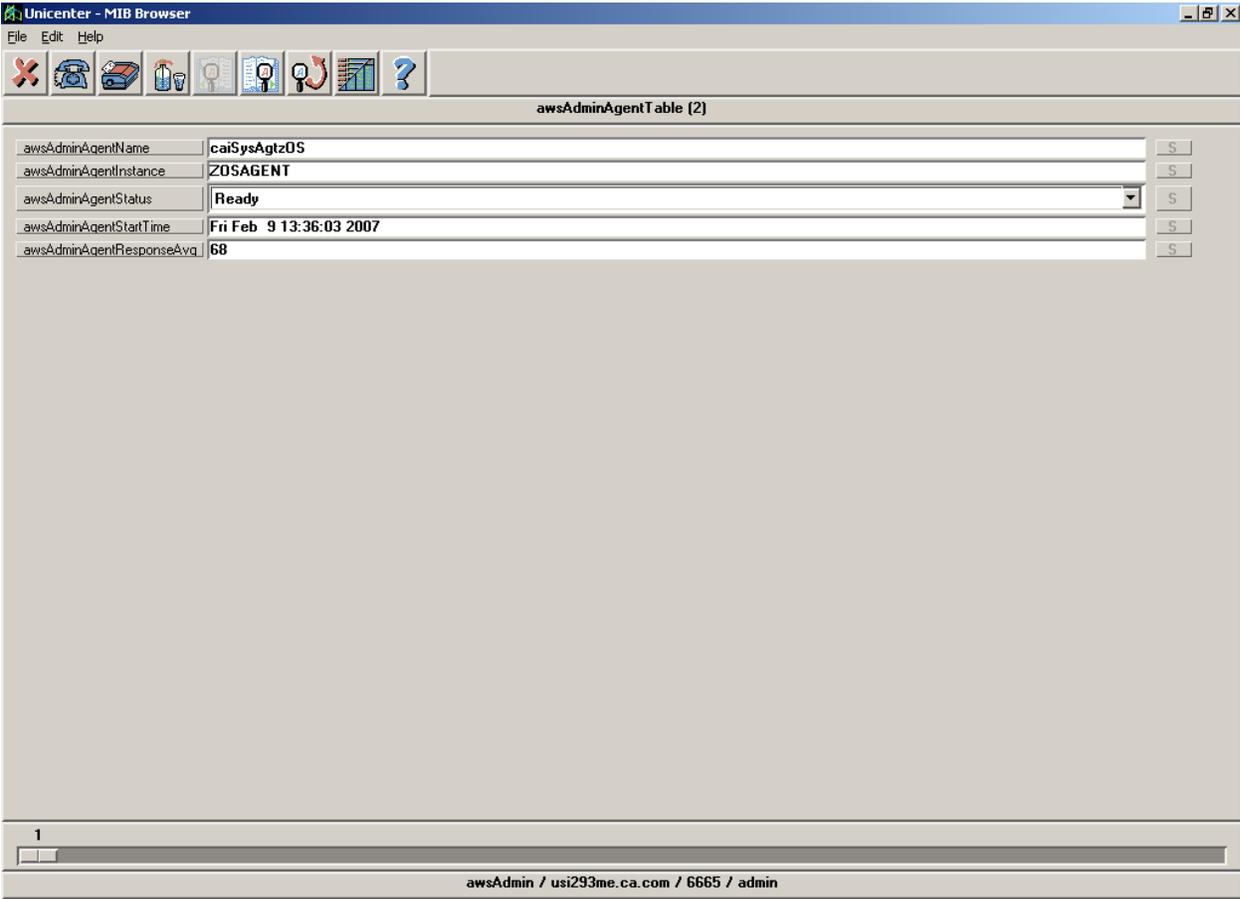
If not, no MIB-mixed agents will be discovered on this node.

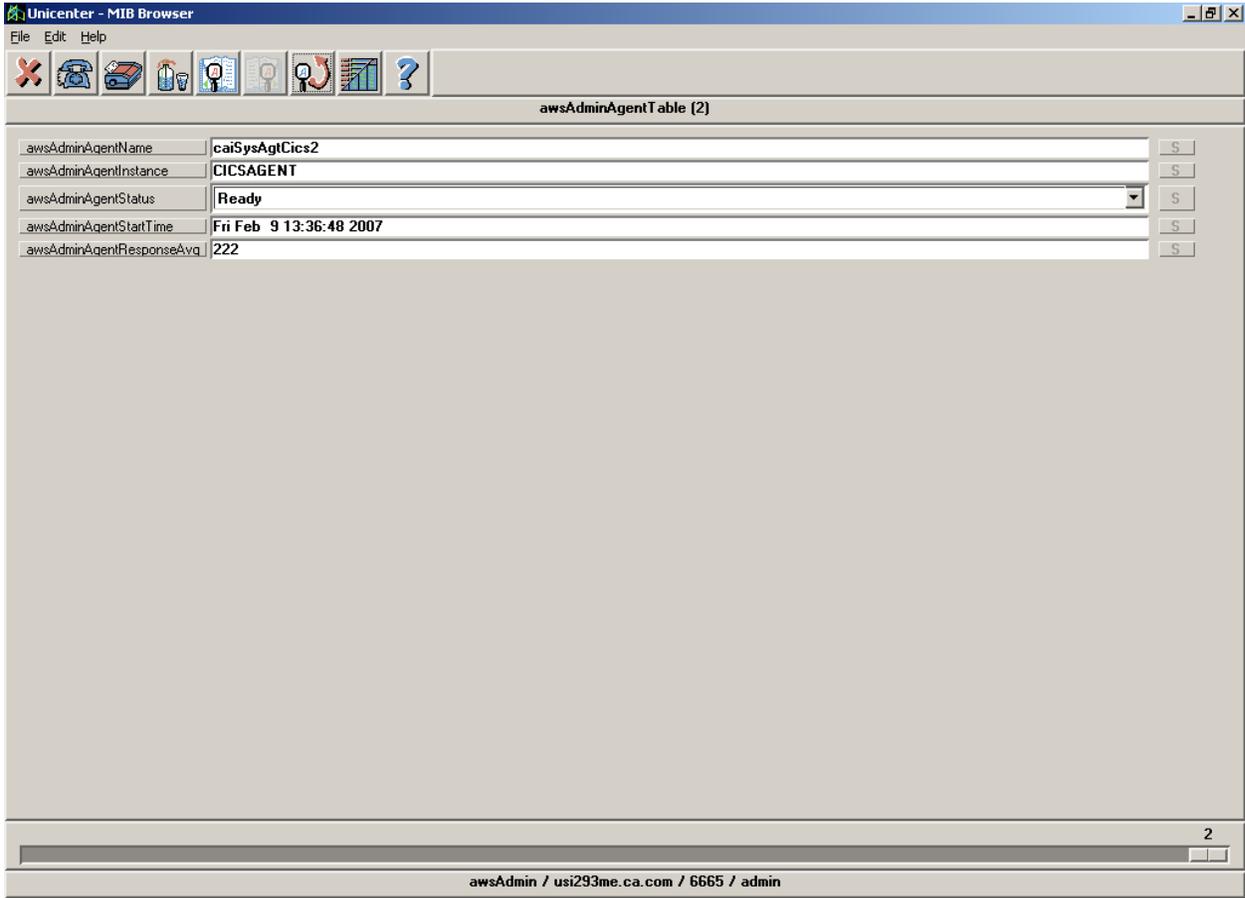
- Can you Mibbrowse the awsAdmin MIB on the mainframe?

The second-level discovery process uses the awsAdmin MIB to discover agents running on z/OS.

1. Run mibbrowse in DOS from the NSM machine to verify this capability.
2. Connect to the awsAdmin MIB on the appropriate host name and UDP port (6665 by default).
3. Attempt to drill down though awsAdminAgentGroup >awsAdminAgentTable.

You should see each active agent listed. For example:





Performance Considerations

This section describes performance considerations for Agent Technology for z/OS.

Adequate CPU

This is a time critical application. To ensure your CA Technologies workstations receive timely responses to their information requests, the necessary software components must receive adequate CPU cycles to process the workload. These components include:

- The TCP/IP region running on the mainframe
- The Agent Technology Services (and OMVS itself)
- Any (all) z/OS based agents

This typically means running these components in a high priority performance group, equal to that of other main systems and started tasks (for example, JES, VTAM, CA Top Secret, and so on).

The Object Store Database

A total of 22 files comprise the `aws_sadmin` object store database. These files (found within directory `$AGENTWORKS_DIR/services/var/aws_sadmin`) can be categorized into the following groups:

- Objects table-6 files prefixed with OBJECTS
- Titles table-6 files prefixed with PROPTH
- Properties table-6 files prefixed with PROPVH
- Votree table-4 files prefixed with VOTREE

The size of these files depends on the following:

- The number of MIBs loaded in the object store
- The size and number of configuration sets loaded in the store
- The number of agent instances currently running

To minimize the number of EXCPs used to access the object store, the following parameters have been added to the `aws_sadmin` startup command. The startup of the `aws_sadmin` service can be found in file `awsservices.cfg`, within directory `services/config/awsservices`.

```
-x <cache-size for the objects table>  
-y <cache-size for the properties table>  
-z <cache-size for the votree table>
```

Note: No parameter has been established for the Titles table because the default value should be adequate for all sites. All cache sizes are expressed in KB. When left unspecified, the default value for all three parameters is 1024.

Determine the Best Values for the Three Caches

To minimize I/O EXCPs to the object store database, you must set the cache size parameters to the total size of all of the files within each of the different table categories (with the exception of the rollback files, which are suffixed with MMA, RBI, MM2, and RB2). The following are the calculations you should perform to determine the size for each of the cache parms:

-x

Sum the size of the following files (divide the result by 1024 to convert to KB):

- OBJECTS.CHK
- OBJECTS.FBM
- OBJECTS.MDI
- OBJECTS.MDS
- Sum of OBJECTS file sizes in bytes / 1024

-y

Sum the size of the following files (Divide the result by 1024 to convert to KB):

- PROPVH.CHK
- PROPVH.FBM
- PROPVH.MDI
- PROPVH.MDS
- Sum of PROPVH file sizes in bytes / 1024

-z

Sum the size of the following files (Divide the result by 1024 to convert to KB):

- VOTREE.CH2
- VOTREE.CM2
- Sum of VOTREE file sizes in bytes / 1024

Note: These calculations should be performed only after all MIBs have been loaded and all agent instances have been started. You must ensure that your Agent Technology for z/OS Services have been shut down before you attempt to modify the awservices.cfg file (otherwise your changes will not be preserved).

Even if only one agent is active at a time but the store contains all MIBs, the size of the cache for the VOTREE table needs to be at least the same as the size of its corresponding file. A smaller value (even within a few KB) leads directly to an increase of EXCPs.

This is not the case for the two other tables, although the best size for the corresponding caches needs to be higher than the size really occupied by the corresponding MIB definition, plus the size used by all the instances of the agent. Therefore, we recommend you allocate them as the total size of the different tables.

Customize Cache Sizes

You can customize the table cache sizes using *either* of the following procedures.

To customize the table cache sizes using `install_agents` script file

1. Edit the `$AGENTWORKS_DIR/services/tools/install_agents` script file to update the `x`, `y`, `z` values on the `servicectl` command related to `aws_admin` with your new values.
2. Run the `install_agents` script with the "remove `aws_admin`" parameters.
3. Rerun the `install_agents` script with the "installauto `aws_admin`" parameters.
Agent Technology uses the updated cache sizes.

To customize the table cache sizes using the `awsservices.cfg` configuration file

Note: To use this procedure, you are required stop Agent Technology.

1. Stop Agent Technology.
2. Edit the `$AGENTWORKS_DIR/services/config/awsservices/awsservices.cfg` configuration file to update the `x`, `y`, `z` values of the `aws_admin` entry with your new values.
3. Restart Agent Technology.
Agent Technology uses the updated cache sizes.

Configuration

You can change the way agents handle information on two levels:

- Agent level
- Event manager level

Agent Level

At the agent level, you can change the aspects of the resource you are monitoring, how often you gather information from the resource, how often you report the information to the manager, and so on. For example, with the z/OS system agent, you can specify that file systems are to be monitored and the exact name of the file systems to be monitored for this computer.

Agent level configuration has the following parts:

- MIB that defines the attributes of the managed objects
- Configuration sets that define which objects the agent monitors, how the agent gathers resource data, and how the agent assesses the state of managed objects

Note: For more information about configuring agents, see the CA NSM documentation.

Event Manager Level

At the event manager level, you can change the actions to take in response to events or traps sent by agents.

Note: For more information about configuring event actions, see Correlation of Event Console Messages in the chapter “Event Management.”

Turn off Statistics Collection

The `aws_sadmin` service is responsible for processing all SNMP requests coming in from the various CA Technologies components and for the generation of SNMP traps. In addition to these functions, the `aws_sadmin` service also maintains the `awsAdmin` MIB and the associated `awsAdmin` agent. The `awsAdmin` MIB contains the following categories of information:

- Configuration information
- Agent and MIB information
- Statistical information

The statistical information is collected in the following different groups:

- `AwsAdminAgentGroup`
- `AwsAdminSnmpGroup`
- `AwsAdminPerfGroup`
- `AwsAdminSourceGroup`

The maintenance of the statistical information within the awsAdmin MIB represents an appreciable amount of work for the aws_sadmin service. In fact, benchmark tests have shown that as much as 40% of the CPU time spent by the aws_sadmin process is consumed by the updating of all of these statistics.

As a performance-enhancing feature, the following environment variables have been added to control the collection of the statistical information within the awsAdmin MIB. These environment variables let you turn off the collection of any (all) statistics, resulting in the reduction in CPU (as well as increased performance) for the aws_sadmin service. The environment variables, along with their possible values, are as follows:

```
AW_ADMIN_STAT_AGENT=ON/OFF
AW_ADMIN_STAT_SNMP=ON/OFF/NOTOTAL
AW_ADMIN_STAT_PERF=ON/OFF
AW_ADMIN_STAT_SOURCE=ON/OFF
```

The default value for each variable is ON. If you want to change this value (to improve performance and reduce CPU overhead), define the environment variable within the agentworks.profile file in the root Agent Technology directory (\$AGENTWORKS_DIR) of your zFS.

Integration with CA OPS/MVS Event Management and Automation

Agent Technology services generate the following messages at startup and shutdown. These messages are useful for integrating with CA OPS/MVS Event Management and Automation for automating the Agent Technology startup and shutdown processes.

```
UAT0001I awservices is started
UAT0002I awservices is stopped
UAT0011I Starting agent_name:instance_name
UAT0012I agent_name:instance_name is stopped
UAT0013W agent_name:instance_name is status reason : reason
```

The following messages trigger events for individual agents:

```
UAT0101I Agent agent_name is ready
UAT0102I Agent agent_name is stopped
```

To control the message destination, you set the `AW_MESSAGE_OUTPUT` environment variable in both your `CAW0OPTV(ENVFILE)` member and your `agentworks.profile` file (in the Agent Technology root directory) to one of the following values:

- `WTO`—Operators console
- `SYSLOG`—System log (default)
- `BOTH`—Both operators console and system log
- `NONE`—None

Chapter 4: Resource Initialization Manager

This section contains the following topics:

[Overview](#) (see page 93)

[Features](#) (see page 93)

[How the Process Works](#) (see page 94)

[Operation](#) (see page 96)

[Error Handling](#) (see page 99)

[IF/ENDIF Logic Statements and System Symbols](#) (see page 100)

[Verification Utilities](#) (see page 104)

[CA LMP](#) (see page 108)

[CAISSF](#) (see page 125)

[Reporting Licensed Registered Product Usage](#) (see page 125)

[CAS9INIT CAIRIM Initialization Routine](#) (see page 130)

[CAMODID](#) (see page 133)

Overview

The Resource Initialization Manager (CAIRIM) service prepares your operating system environment for all of your CA Technologies products and then starts them. CAIRIM lets you install CA Technologies products dynamically. You do not need to install SVCs, link pack area (LPA) modules, subsystem definitions, SMF exits, or make other permanent changes to your operating system setup.

CAIRIM also features two subcomponents, CA LMP for license management and CAISSF for security management. These are discussed in detail at the end of the chapter.

Features

CAIRIM simplifies the installation of CA Technologies products and provides improved reliability, availability, and serviceability. This service provides the following features:

- Installation of z/OS Interfaces—CAIRIM eliminates the need for user SVCs, LPA modifications, SMF exits, or additions to your subsystem table. It installs the interfaces to z/OS that are required by CA Technologies products and common components.

For products that require SMF information, the CAI SMF Interceptor (CAISMFI) dynamically processes the SMF data independent of any SMF exits or SMF control settings.

- Verification of Installation—During initialization, CAIRIM verifies that all requested product services are satisfied. After initialization, you can use verification utilities to review the initialized CA Technologies product interfaces.

- **Automatic Startup of Software**—CAIRIM supports an auto commands function for issuing commands upon completion of the CAIRIM initialization processes. You can issue auto commands regardless of the success of any initialization process, and you can use them to start any product or task in your system.
- **Proper Timing and Order of Installation**—CAIRIM provides the basic environment needed for all CA Technologies products prior to the initialization and startup of the products themselves. Multiple CA Technologies products share a single initialization request for each component.

For CA Technologies products that must obtain control before the job entry subsystem (JES) starts, CAIRIM must be executed before JES2 (SUB=MSTR).
- **Installation and Maintenance of Products Without an Initial Program Load (IPL)**—CAIRIM can install a product without an IPL. You can run CAIRIM to install new products without affecting those already installed on your active system.

Because CA Technologies products can use CAIRIM services to re-initialize operating modules, you can apply emergency maintenance, and if necessary, remove it from your active system, without an IPL.

Specification of different program libraries and initialization routines for testing and backout is also supported. Your usage of this feature depends on the level of support provided by the CA Technologies product you are using.
- **Comprehensive Error Handling**—CAIRIM processes abends or other errors in any initialization routine, and issues messages for them. This makes problem detection and handling easier than that encountered with a disassociated set of system modifications.

How the Process Works

CAIRIM is parameter driven, accepting input from a sequential file allocated by a PARMLIB DD statement in the CAWOPROC(CAS9) member. The file contains product-specific parameter statements. CAIRIM initializes the product requirements specified in the parameter statements, followed by any requested auto commands.

Parameter Statements

If a product requires a parameter statement, the installation documentation provides the information.

Each parameter statement specifies a product, version, the level, and other optional parameters.

Example:

```
PRODUCT(CA-LIBRARIAN) VERSION(LJ43) INIT(LJ43INIT)
```

Note: For more information about how to use parameter statements, see the *Installation Guide*.

Initialization Routines

You specify the name of an initialization routine through the INIT parameter. CAIRIM validates this routine against specified parameters and then executes the routine. The initialization routine in turn invokes various CAIRIM facilities to install the operating system interfaces, such as SVC routines, LPA modules, and SMF intercepts. You can put the product initialization routine in a link list (LNKLST) data set, the CAIRIM STEPLIB, or a private library.

Initialization Data Sets

You can direct CAIRIM to a specific data set for loading the initialization routine by specifying the LOADLIB parameter. The data set must be APF authorized. The LOADLIB parameter causes TASKLIB for the routine to switch to that data set.

If LOADLIB is used, the initialization routine and the associated CAIRIM program modules must be either in the LINKLST or in the data set specified by the LOADLIB parameter. If they are only in a CAIRIM STEPLIB, an S806 abend will occur.

Auto Commands

After invoking the initialization routines, CAIRIM issues any auto commands that you specified in the [auto commands member](#) (see page 98) defined by the AUTOCMDS DD statement. To prevent dual startups, these commands are issued only during the first time CAIRIM is run after each IPL. Subsequent runs of CAIRIM result in the execution of the pending initialization routines, but the auto commands are not reissued.

For CAIRIM to reissue auto commands, you can use the AUTOCMDS parameter when you rerun it.

Verification of Initialization

CAIRIM stores the result of each initialization routine in the form of date, time, and success or failure based on the return code. The CAIRIMU, CAISFMU, and CAISUBU [verification utilities](#) (see page 104) display this information. If you experience problems, use these utilities to check the initialization.

Operation

You can run CAIRIM any time prior to the operation of the CA Technologies product that depends on it.

Note: It is recommended that you do not start CAIENF until after CAIRIM ends.

If necessary, you can run CAIRIM multiple times because it tracks products that are installed correctly and bypasses the initialization of those products that are already running.

You can run CAIRIM in one of the following ways:

- As a started task
- Under the master subsystem
- As a batch job
- (Recommended) Automatically at IPL

Run CAIRIM as a Started Task

You run CAIRIM as a started task unless an installed CA Technologies product requires CAIRIM to be run before JES.

To run CAIRIM as a started task

1. Make a copy of the CAW0PROC(CAS9) member.
2. Update the copy to suit the requirements of your site.
3. Move the updated copy to a system proclib.
4. Issue *one* of the following console commands:

```
START CAS9
```

```
START CAS9,, ,AUTOCMDS
```

```
START CAS9,, ,EKGP
```

AUTOCMDS

Reissues all the commands specified by the AUTOCMDS DD statement.

Note: Usually, CAIRIM only issues auto commands the first time it is run. This parameter forces auto commands to be reissued on subsequent startups.

Important! Before using the AUTOCMDS parameter, you should check that reissuing the commands do not cause problems. The parameter can result in duplicate started tasks.

EKGP

Causes CAIRIM to prompt the operator to input a valid EKG LMP Key.

5. CAIRIM starts.

Run CAIRIM Under the Master Subsystem

You run CAIRIM under the master subsystem if an installed CA Technologies product requires CAIRIM initialization before JES.

To run CAIRIM under the master subsystem

1. Ensure that a copy of the CAW0PROC(CAS9) member is updated to suit the requirements of your site, is in a system proclib.

Note: When running CAIRIM under the Master subsystem, you cannot direct SYSPRINT to SYSOUT=*. You must specify a SYSPRINT DD DUMMY statement or, if you want the SYSPRINT output, direct SYSPRINT to a permanent data set.

2. Use the following command to start CAIRIM before JES:

```
START CAS9,SUB=MSTR,TIME=1440
```

CAIRIM starts under the master subsystem.

Run CAIRIM as a Batch Job

You run CAIRIM as a batch job to retain the output.

To run CAIRIM as a batch job

1. Make a copy of the CAW0JCL(CAIRIM) member.
2. Update the copy to suit the requirements of your site.
3. Submit the copy as a job.

CAIRIM starts.

Automate the Startup of CAIRIM at IPL

To automate the startup of CAIRIM at IPL

1. Add the following command to the SYS1.PARMLIB(COMMNDxx) member that is used for the IPL.

`COM='START CAS9' CAI RESOURCE INITIALIZATION MANAGER`
2. Move the z/OS auto commands that you want to issue after CAIRIM (for example, the START JES command if you want to start JES after CAIRIM) from the SYS1.PARMLIB(COMMNDxx) member to the auto commands member defined by the AUTOCMDS DD statement for CAIRIM.

At the next IPL, CAIRIM will start automatically.

Multiple members can be concatenated for processing and it does not matter which member contains the command.

CAIRIM Auto Commands Member

After the initialization of the CA Technologies products, irrespective of whether initialization was successful, CAIRIM reads the member defined by the AUTOCMDS DD statement and issues each command in the member.

The CAIRIM auto commands member specifies one command per line, exactly as it would be issued from the operator console. You can add comments by placing an asterisk (*) in Column 1. You can also add an option, WAIT(*nnn*), which causes the auto command process to pause *nnn* seconds, where *nnn* is a number between 0 and 999.

Example: Auto Command for Starting CA Scheduler Job Management

```
Col 1  
  ↓  
  * Bring up CA-Scheduler  
  START CAOMS  
  WAIT(30)
```

Error Handling

The following error conditions are handled by CAIRIM:

Failure to allocate, open, or read the parmlib data set

Processing continues to issue the CAIRIM auto commands.

Syntax errors in control cards

Processing continues with the next PRODUCT parameter.

Errors while allocating, opening, or reading the product load library

Processing continues with the next PRODUCT parameter.

Missing product initialization routine

Processing continues with the next PRODUCT parameter.

Abends or nonzero return codes from any product installation routine

Processing continues with the next PRODUCT parameter.

Error while opening or reading the CAIRIM auto commands member

Processing terminates.

General abend anywhere within the CAIRIM routine

CAIRIM tries to associate the abend with a specific product initialization function and terminates that function. If CAIRIM cannot isolate the error, it aborts any further dynamic product installations. In either case, CAIRIM issues the auto commands.

SVC dumps by CAIRIM have the following title:

```
CAIRIM INITIALIZATION DUMP. PRODUCT = product VERSION = version
```

product and *version* identifies the product being initialized. If the error cannot be isolated to a single product, *product* and *version* are replaced by N/A.

When a product initialization routine ends with a nonzero return code, the CAS9140E error message is issued. By default, this message is not highlighted. If you want to have this message highlighted and make it non-scrollable, add this statement in the RIMPARM member defined by the PARMLIB DD statement, starting in Column 1:

```
HIGHLIGHT=ON
```

The statement affects the product initialization routines that follow it. To turn off this behavior later in the member, add the following statement:

```
HIGHLIGHT=OFF
```

IF/ENDIF Logic Statements and System Symbols

With PTF xyz, CAIRIM offers the following for RIMPARMs for improved SYSPLEX parm sharing capability.

- IF/ENDIF
- LOG/NOLOG
- System symbol substitution

For example, you could code a data set name that includes system symbols on the RIMPARM LOADLIB optional keyword such that an APF failure on such a LOADLIB would let CAIRIM continue to the next RIMPARM statement.

ENDIF

ENDIF is a keyword that is used to terminate a corresponding IF keyword statement. ENDIF keywords must equally match the number of IF keyword statements specified in a parameter file. If you have more ENDIF keyword statements than there are corresponding IF keyword statements, a syntax error occurs. The statement on which the error was found is returned.

ENDIF is only recognized as a keyword if it is the first verb read in by the Common Parameter File Reader Service on a given input statement. You may use comments on an ENDIF keyword statement. However, if extraneous data is found on a keyword statement, a syntax error occurs. The statement on which the error was found is returned.

If an ENDIF keyword statement is successfully processed, the statement is not returned to the caller. Conversely, the service begins reading the next statement.

IF

The IF keyword statement is used to support "if" logic in a parameter file. Several cases exist for when a syntax error can be found on an IF keyword statement. For all syntax errors, the statement on which the error was found is returned.

The IF keyword takes on the form:

```
IF &symbol. [!]= (value0 [,value1] ...)
```

The IF keyword statement is only recognized as a keyword if it is the first verb read in by the Common Parameter File Reader Service on a given input statement. One or more spaces must follow this keyword. If a space does not follow "IF," it is not treated as a keyword. IF keyword statements can be nested up to 12,287 levels.

If the service determines that the IF keyword statement is present, and the caller of the service specified SYMSUB=NO on the OPEN call, symbol substitution is done on the next data item found, &symbol., unconditionally.

Symbol substitution

&symbol.

Specifies a data item to compare against the values on the right side of the compare operator. It will have been passed to the #SYMSUB service. If it is found to be a defined symbol, it is substituted to its appropriate value (for example, if &symbol. was &SYSNAME., after substitution, &SYSNAME. is the name of the current system after substitution). &symbol. One or more spaces must follow &symbol. If it is not followed by a space, a syntax error occurs.

[!]=

Identifies the compare operator. Valid operators are '=' and "!=" which are interpreted as "equal" and "not equal" respectively. If the operator is '=', the &symbol. is compared against the values on the right side of the compare operator and the IF keyword statement is evaluated as true if the &symbol. is equal to any compare values. If the compare operator is "!=", the &symbol. is compared against the values on the right side of the compare operator and the IF keyword statement is evaluated as true if the &symbol. is not equal to any compare values. One or more spaces must follow [!]=. If it is not followed by a space, a syntax error occurs.

(value0 [,value1] ...)

Identifies the values to compare against the &symbol. based on the specified compare operator. This part of the IF keyword statement is parsed with the Parse and Scan Facility (PSF) Common Service. Up to 32 different compare values can be specified here. If more than one value is to be compared, enclose the values inside of parentheses and separate each value by a comma. If PSF finds a syntax error and an error message buffer is supplied by the caller on the READ request, an error message is returned in the error message buffer.

If any part of the IF keyword statement as described in its form is omitted, a syntax error occurs.

If an IF keyword statement is successfully processed, the statement is not returned to the caller; rather, the service begins reading the next statement.

LOG/NOLOG Keywords

LOG and NOLOG are keywords used to control logging attributes of a parameter file.

When the LOG keyword is in use, each statement image in the parameter file is written to the system log. Each statement image in this case is prefixed by the message identifier CAMS504I. Additionally, if the statement image contains a symbolic value that has been successfully substituted, the statement image that contains the substituted value is also written to the system log. Each statement image in this case is prefixed by the message identifier CAMS505I.

Note: When reading from the Logical Parmlib Concatenation, statement images that contain symbolic values are only logged with the substituted values.

When the NOLOG keyword is in use, no records in the parameter file are written to the system log.

LOG and NOLOG keywords may be used any number of times in a parameter file. The current log specification remains in effect until a different log specification is requested. The current log specification also remains in effect for parameter files that are opened with the INCLUDE keyword. Changes in the log specification in a parameter file that was opened with the INCLUDE keyword only remain in effect until the INCLUDE file is closed. After an INCLUDE file is closed, the logging attributes for the previous member carry on.

LOG and NOLOG are only recognized as keywords if they are the first verbs read in by the Common Parameter File Reader Service on a given input statement. You may use comments on the LOG and NOLOG keywords statements. However, if extraneous data is found on a keyword statement, a syntax error occurs and the statement on which the error was found is returned.

If a LOG keyword statement is successfully processed, the effect of the keyword takes effect immediately.

If a NOLOG keyword statement is successfully processed, the effect of the keyword takes effect beginning with the next statement image read in.

Note: In the following examples, assume that all statements are read while executing on a system name "SYS1" with the "z/OS" operating system.

LOG Example

For the following example, assume that NOLOG is in effect at the time the first statement is read. The following input statements use LOG and NOLOG keywords:

```
LOG
VERB1 KEYWORD1 = (&SYSNAME.)
NOLOG
VERB2 KEYWORD2 = (OPERAND2)
Would be logged to the system log as:
CAMS504I LOG
CAMS504I VERB1 KEYWORD1 = (&SYSNAME.)
CAMS505I VERB1 KEYWORD1 = (XE97)
CAMS504I NOLOG
```

Equal Example

For this example, the following input statements use an "equal" compare:

```
IF      &SYSNAME. = SYS1
        ECHO=YES
ENDIF
```

These statements cause echoing to be turned on for all proceeding RIMPARM statements, if the statement evaluates as true. ENDIF must be paired with an equal number of IF keyword statements.

Not Equal Example

For this example, the following statements use a "not equal" compare:

```
IF    &SYSNAME. != SYS1
      SMPPARM = ON
      LMPEXIT = YES
ENDIF
```

These statements skip the SMPPARM=ON statement and the LMPEXIT=YES statement because &SYSNAME. is substituted as "SYS1", and the "if not" statement evaluates as false. Skipped statements are processed but not returned to the caller on a READ request.

Nested IF Statements Example

For this example, the following statements use nested IF keyword statements:

```
IF    &OSNAME. = z/OS
      ECHO = YES
      IF &SYSNAME. = (SYS1, SYS2)
        LOG
        SMPPARM = ON
      ENDIF
      IF &SYSNAME. != (SYS1, SYS2)
        NOLOG
        LMPEXIT = YES
      ENDIF
ENDIF
```

These statements cause echoing to be turned on if the operating system's name is z/OS. If the &SYSNAME. is equal to SYS1 or SYS2, logging is turned on, and SMPPARM=ON is processed. If the &SYSNAME. is not equal to SYS1 or SYS2, logging is turned off, and LMPEXIT=YES is processed.

Symbol Substitution Example

For this example, the following statements use symbol substitution within a LOADLIB parm:

```
PRODUCT(CA DATACOM) VERSION(BD12) INIT(DBRIMPR) -
LOADLIB(&LOADHLQ. .DCR1200Z.SMPE.CAAXLOAD) -
PARM(D247,DBSVCPR,TYP=3)
```

These statements cause the value for &LOADHLQ. to be substituted. The resulting statement must follow z/OS conventions for Partition Data Set(PDS) qualifier names. The resulting PDS is then checked for a member coded on the INIT statement.

Verification Utilities

You can use the following utilities to verify the initialization of your resources:

- CAIRIMU displays the status of all modules initialized by CAIRIM.
- CAISMFU displays the status of the SMF exits installed by CAIRIM.
- CAISUBU displays the status of the subsystems installed by CAIRIM.

Operation

You can execute the CAIRIM utilities in one of the following ways:

- As a batch job—The CAW0JCL data set contains the sample JCL members for invoking each utility from a batch job.
- As a started task—The CAW0PROC data set contains the sample members.
- From a TSO CLIST—Sample CLISTS are provided in the CAI.CAW0CLS0 utilizing LIBDEF.

Note: When you execute the utilities as started tasks or batch jobs, the output is in the form of WTO messages sent to the system console or the batch job log. In TSO, messages are sent directly to the TSO session for the user.

CAIRIMU Utility

The Resource Initialization Manager Utility, CAIRIMU, lists the product or component name, version and level, and displays the result of each initialization routine in the form of date, time, and success or failure based on the return code. By executing this utility, you can determine which products and components have been initialized.

Example: Typical Output

Generally, the output from CAIRIMU is as follows:

```
CAS9006I THE FOLLOWING PRODUCTS HAVE BEEN INITIALIZED:
CAS9002I
CAS9007I PRODUCT          VERS      INIT    DATE/TIME
CAS9008I  product          version   init    date time
          . . .
CAS9009I INITIALIZED FROM loadLib
CAS9002I
CAS9010I NUMBER OF INTERFACES: nn
```

Note: For information about messages, see the *CA Common Services Message Reference Guide*.

Example: Possible Error Output

The following output indicates an error:

```
CAS9011I    NO CAI PRODUCT INTERFACES PRESENT
```

If CAIRIM is not expected to initialize any products upon startup, this is not an error.

If one or more of your products require CAIRIM initialization, it indicates that CAIRIM has not been run or has encountered a problem before initializing any products. You should run or rerun CAIRIM after correcting any problems.

Example: CA LMP Output

If CA LMP has encountered problems, the following output from CAIRIMU can appear:

```
CAS9012A    nnnn License warnings/violations on CPU @@@@
CAS9013A    Product %% expired and is still in use.
```

This output means that CAIRIM has initialized the product, but CA LMP noted a warning or violation. The product is permitted to run, but messages are issued if the problem is not corrected.

For assistance, contact CA Total License Care (TLC). For more information, see Contact Technical Support.

CAIRIMU PROD Parameter (Optional)

You can use the optional PROD parameter with the CAIRIMU command to view detailed licensing information about your installed CA Technologies products.

This parameter has the following format:

```
CAIRIMU PROD[V][{(xx-yy)}
```

V

Designates that the list generated contains only those products that are in violation or approaching the product expiration date.

xx and yy

Designates the product range.

Example: Display all products

To display extended usage/status information for all products, use the PROD parameter by itself.

```
CAIRIMU PROD
```

Example: Display a single product

To display extended usage/status information for a single product enter the PROD parameter with one product in the range. This example displays usage/status information for product A1.

```
CAIRIMU PROD(A1)
```

Example: Display a product range that are in violation or near expiration

To display extended usage/status information for a product range enter the PROD parameter with two products in the range. This example shows products in the range A1-A9 that are in violation or near expiration.

```
CAIRIMU PRODV(A1-A9)
```

CAISMFU Utility

The Interceptor Utility, CAISMFU, displays the status of the SMF exits installed by CAIRIM. The display includes the product associated with the SMF module, the version of the module, the name of the module, the purpose of the module, and the status of the module.

Example: Typical Output

Generally, the output from CAISMFU is as follows:

```
CAS9001I  INTERFACE SUMMARY:
CAS9002I
CAS9003I  NAME      VERS   DESCRIPTION  STATUS
CAS9004I  pgmname  caid   caidesc    caistat
CAS9002I
CAS9005I  NUMBER OF INTERFACES: nn   NUMBER OF CALLS PROCESSED: nn
```

Note: For information about messages, see the *CA Common Services Message Reference Guide*.

Example: Possible Error Output

The following output indicates a possible error:

```
CAS9011I  NO CAI PRODUCT INTERFACES PRESENT
```

If none of your products install SMF exits using CAIRIM, this is not an error.

If one or more of your products install SMF exits using CAIRIM, it indicates that CAIRIM has not been run or has encountered a problem. You should run or rerun CAIRIM. If the problem persists, contact CA Support.

Note: For information about the use of SMF exits by a product, see the CAIRIM requirements in the product installation guide.

CAISUBU Utility

The Subsystem Utility, CAISUBU, lets you display the status of the subsystems installed by CAIRIM. The utility verifies the current status of the subsystems, their subsystem control table address, and their program initialization programs, if any.

The status display includes the four-character subsystem name, the SSCT address (of an internal z/OS data structure), the status of the subsystem, and the subsystem initialization program, if any.

Example: Typical Output

Generally, the output from CAISUBU is as follows:

```
CAS9040I  SUBSYSTEM SUMMARY:
CAS9002I
CAS9041I  NAME          SSCT ADDRESS  STATUS      INIT ROUTINE
CAS9042I  subname          @@@@      caistat    initname
CAS9002I
```

Note: For information about messages, see the *CA Common Services Message Reference Guide*.

Example: Possible Error Output

If CAIRIM has not been run or has encountered a problem, the following output from CAIRIMU appears:

The following output indicates a possible error:

```
CAS9011I  NO CAI PRODUCT INTERFACES PRESENT
```

If none of your products install subsystems using CAIRIM, this is not an error.

If one or more of your products install subsystems using CAIRIM, it indicates that CAIRIM has not been run or has encountered a problem. You should run or rerun CAIRIM. If the problem persists, contact CA Support.

CA LMP

CA LMP (License Management Program) is a subcomponent of CAIRIM that provides a standardized and automated approach to the tracking of licensed software. It uses enforcement software to validate the configuration and to report on activities regarding the licensing, usage, and financial aspects of CA software.

How CA LMP Works

CA LMP is designed to operate smoothly and efficiently, whether you are using one CA solution on one central processing unit (CPU) or multiple CA solutions on several CPUs.

Each computer that is running one or more CA Technologies products maintained by CA LMP uses common enforcement software and a common KEYS data set member. The KEYS member contains the CA LMP execution keys that are required to run the associated CA solutions on each of the specified CPUs. During product installation, you transfer the execution keys from the CA LMP Product Key Certificates to control statements in the KEYS member.

CA LMP is executed as part of the CAIRIM service. The KEYS DD statement in the CAS9 procedure points to the KEYS member. When CAIRIM is started, each statement is read and verified by CA LMP in sequential order, as found in the KEYS member.

During the operation of each CA solution, the CA LMP enforcement software is invoked periodically. This software compares the execution keys with the execution environment. If there is a discrepancy between the execution keys and the environment, it issues messages so that you can resolve the situation and avoid any interruption in the operation of the solution. If after 24 hours there are no further violations for a CA solution, the messages will cease.

The messages are written to the system console, the Event Console, a batch job log, or a TSO terminal, providing a history of the warnings.

After the messages are issued, the solution continues operation. The enforcement software ensures that a solution under the control of CA LMP is not interrupted because of expiration dates, improper execution keys, or changes in the CPU on which it is running.

Example: Verification of Execution Keys at CAIRIM Startup

Each control statement is displayed as it is read, as shown in the following sample:

```
CAS9075I - SERVICE(CA-RIM/BASE ) VERS(1200) GENLVL(0808AW000)
CAS9115I - INPUT: *
CAS9115I - INPUT: * KEY PARAMETERS FOR LMP
CAS9115I - INPUT: *
CAS9115I - INPUT: PROD(CH) CPU(3090-600 /000000) DATE(19JUL11) ❶
      LMPCODE(XXXXXXXXXXXXXXXXXX)
CAS9190I - PRODUCT CH KEY ACCEPTED FOR THIS CPU ❷
CAS9115I - INPUT: PROD(S0) CPU(3090-600 /000000) DATE(31OCT10) ❸
      LMPCODE(YYYYYYYYYYYYYYYY)
CAS9125E - INVALID DATA: KEY ALREADY EXPIRED ❹
CAS9115I - INPUT: PROD(S0) CPU(3090-600 /111111) DATE(30DEC12) ❺
      LMPCODE(ZZZZZZZZZZZZZZZZ)
```

The statements give the following results:

- The execution key in ❶ is valid. The message in ❷ indicates that the key is accepted.
- The execution key in ❸ has expired. The message in ❹ displays the fact. A similar situation would exist if the key is invalid or has been tampered with.
- The execution key in ❺ is for a CPU other than the one for which CAIRIM is attempting to initialize solutions. The key is displayed, but no message follows and no action is taken. This can occur if you are sharing a common member to define the execution keys for multiple CPUs.

Product Execution Key Control Statements

Depending on the type of license, a product execution key statement can have the following formats:

```
PROD(pp) DATE(ddmmyy) CPU(tttt-mmmm/ssssss) LMPCODE(kkkkkkkkkkkkkkk)
```

```
PROD (pp) CPU (tttt-mmmm/ssssss) BEGINS (ddmmyyyy) EXPIRES (ddmmyyyy)
      LMPCODE (kkkkkkkkkkkkkkkk) [DELETE | REMOVE]
```

PROD

Specifies the two-character product code.

DATE

Specifies the expiration date of the CA LMP licensing agreement. Date supports the twenty-first century date format, and only requires the last two digits of the year. For example, January 1, 2010 is 01JAN10, with 10 representing the year 2010.

CPU

Specifies the type, model, and serial number of the CPU (for example, 9672-R83 /447276) on which the software solution is to run. If the type and model require less than four characters, blank spaces are inserted at the end of each field for the unused characters.

LMPCODE

Specifies the execution key needed to run the software solution.

BEGINS

Specifies the effective starting date of the key. The key is not valid before this date. Use four digits to identify years.

EXPIRES

Specifies the expiration date CA LMP licensing agreement. Use four digits to identify years.

DELETE | REMOVE

Causes the product key to be marked for bypass and therefore skipped on subsequent scans of the Product Descriptor Table (PDT). The product key entry is not physically removed from the PDT, simply rendered inactive with the raising of the bypass semaphore.

This keyword is valid for existing product keys only. A DELETE keyword on a new product definition statement for which a PDT entry does not exist, is rejected and CAIRIM immediately terminates in error.

Note: Key deletion (removal) is effective only if there are no subsequent license checks (#FLOID) issued against the product. If a #FLOID request is issued against a previously removed key, it preempts any previous DELETE action and LMP key violation warnings and alerts will be resumed for the product.

Add Product Execution Keys

If a CA Technologies product uses CA LMP, you receive a Key Certificate with the product installation or maintenance media. The certificate contains the product execution key. To ensure proper initialization of the product, you must add the key to CAIRIM.

To add an execution key to CAIRIM, add its control statement to the CAWOOPTN(KEYS) data set member.

Example: Control Statement

```
PROD(S0) DATE(01JAN10) CPU(9672-R83 /447276) LMPCODE(52H2K06130Z7RZD6)
```

SITEID Type LMP Keys

SITEID type LMP keys are for sites that have selected Audit Governance to manage and control the use of CA software in their organizations.

CAIRIM KEYS dd File Record Coding Conventions

The CAIRIM KEYS dd is a file comprised of 80-byte records. These records may be broken into four logical fields:

- The first is in position 1. If an "*" is found in this field, then the record is treated as a comment.
- The second field is located in positions 2 through 71 of the record, and contains the CALMP Control Statement data.
- The third field is in position 72. If a "-" is found here then this Control Statement is continued on the next record.
- The fourth field is in positions 73 through 80 and may be used for numbering the records.

With the introduction of SITEID processing, it is necessary to allow Control Statements to be broken in the middle of a verb. This is accomplished by coding a " +" (blank space and the plus sign) at the point in the verb data that you wish to break the verb and by adding a "-" (minus sign) in position 72. The data is then continued on the next record.

If you are attempting to break the data in a verb at a space then the space must be included followed by the "+" so that the last 3 characters of the verb look like " +" (2 spaces and a plus) followed by a "-" in position 72.

The Control Statements are not case sensitive, they may be all upper case, all lower case, or a mixture of both.

Examples:

```

1xxxxxxxx1xxxxxxxx2xxxxxxxx3xxxxxxxx4xxxxxxxx5xxxxxxxx6xxxxxxxx7xxxxxxxx8
*
* CALMP KEYS FOR CAIRIM
*
SITEID(00123456) SITECODE(ETH2PHQZTXQPXGXRK7ZPT)
    NAME(SUPER DUPER LONG CLIENT NAME AND THIS ONE IS BIG +
        ENOUGH)
* SITEID(98765432) SITECODE(AFM3XC43BPASTBG0FUUY8)
* NAME(THIS IS A LONG CLIENT NAME)
PRODUCT(S0) DATE(19JUL12)
    CPU(3090-****/ +
        071966) LMPCODE(42E2LZA66ZC7RZDD)
PRODUCT(S0) DATE(12DEC12)
    CPU(3090-****/071966) LMPCODE(22E2LZA +
        663Z7RZDE)
PRODUCT(L0) DATE(12DEC12)
    CPU(3090-****/071966) LMPCoDe(HHGHP8DRRC81T8LG)
EKG(35575167)
PROD(L0) DATE(25DEC11)
    CPU(3090-****/456789) LMPCODE(8HGERW1DC91T8L6)
    
```

```

*   PROD(KO) DATE(05MAR10)                                -00250000
*       CPU(SITE-****/123456) LMPCODE(HEETFYCEP4QED75B)    00260000
PROD(KO) DATE(11NOV12)                                    -00270000
       CPU(SITE-****/123456) LMPCODE(9EE5CYCEPBB8D75T)    00280000

```

Records 00060000, 00150000, and 00180000 show that the operand was continued on another record by coding a " +" (blank followed by a plus sign) as the last 2 characters of the operand on the record to be continued and by adding a "-" in column 72.

Records 00060000, and 00180000 show that in order to break at a blank you must code that blank along with the " +" (blank followed by a plus sign) so that in the Control Statement there is a " +" (2 blanks followed by a plus sign) at the end of the verb data which is followed by the "-" in position 72.

Records 0060000, and 00090000 show that only one SITEID Control Statement may appear in the KEYS dd file, and it must be the first Control Statement aside from comments in the File. The multiple SITEID Control Statements in the example are examples of different formats.

Records 0090000, 00250000, and 00260000 are examples of a commented out Control Statement.

Records 00250000 and 00280000 are examples of a Product Execution Key using SITEID processing.

Proper Ordering of Control Statements within the KEYS dd File

The SITEID control statement must be the first non-commented record in the KEYS dd file. The rest of the records in the KEYS file can be any mixture of Product Execution keys and optionally one EKG Control Statement.

If more than one Product Execution Key for the same product is found and they are valid, then the one with the greatest expiration date is used. For example, if your KEYS dd has two Product Execution Keys for the same product and they are both valid, that is if the current year is 2008 and one KEY expires in 2009 and the other in 2010, then the 2010 KEY is used.

SITEID Control Statement Format

The SITEID control statements are formatted as follows.

Note: Format may be either N for numeric data (0-9) or AN for alphanumeric (0-9, A-Z, or a-z)

Verb	Format	Length	Comments
SITEID	N	8	Contains your CA SITEID (as shown in your SITEID Control Statement).

Verb	Format	Length	Comments
SITECODE	AN	21	Must appear exactly as shown on your SITEID Control Statement.
NAME	AN	80	Customer Name. Must appear exactly as shown in your SITEID Control Statement.

Product Execution Keys Format

The product execution keys have the following format.

Note: When a Product Execution Key is SITEID-specific, the data contained in "CPU" is of the format "(SITE-****/nnnnnn)" where *nnnnnn* is the Site ID which must appear in the SITEID Control Statement without leading zeros. For CPU-specific keys, the format of this field is "(mmmmmmmmm/sssss)" where *mmmmmmmmm* is the CPU model and *sssss* is the serial number.

Verb	Format	Length	Comments
PROD	AN	2	Contains the 2-digit product code for which this key is used.
DATE	AN	7	Clear text date of the earliest possible expiration date of this key.
CPU	AN	16	Contains either the clear text model and serial number (for CPU specific keys) or the Customer SITEID that must appear in the SITEID Control Statement for this key to be valid.
LMPcode	AN	16	Contains data used by CA-LMP.

Load Product Execution Keys

After you add or update a product execution key control statement, you can load it immediately.

To load new keys, execute the CAS9 procedure.

You may want to create a different CAS9 procedure specifically for loading new keys.

Create a Customized CAS9 Procedure for Loading Product Execution Keys

You can create a customized CAS9 procedure to load product execution keys to avoid getting messages about CA Technologies products that were previously initialized using CAIRIM. Although not required, it is recommended that the CARIMPRM member is not read when you execute the CAS9 procedure to load keys.

To create a customized CAS9 procedure for loading product execution keys

1. Make a copy of the CAWOPROC(CAS9) member.
For example, call the copy CAS9LMP.
2. Make the following changes to the copy (CAS9LMP):
 - a. Remove the following line from the PROC statement:
"RIMPARM=CARIMPRM,"
 - b. Remove the following line from the PROC statement:
"AUTOCMD=CAUTOCMD".
 - c. Locate the "KEYS=KEYS," line in the PROC statement and remove the comma as follows:
"KEYS=KEYS"
 - d. Locate the "PARMLIB DD DISP=SHR,DSN=&CAW0OPTN(&RIMPARM)" line and change the line as follows:
"PARMLIB DD DUMMY".
 - e. Locate the "AUTOCMDS DD DISP=SHR,DSN=&CAW0OPTN(&AUTOCMD)" line and modify the line as follows:
"AUTOCMDS DD DUMMY"
3. Move the updated copy to a system proclib.

Load New LMP Keys

To load new LMP keys for your products, start CAS9 with the following parameter:

```
S CAS9, , , LMPKEYS
```

This will not re-init any of your other products and will not re-submit your AUTOCMDS commands.

Reduce CA LMP Console Messages

CA LMP issues messages when processing product execution keys. If you do not want these messages to go to the console, you can reroute the messages to an external flat file using a SYSPRINT DD statement. Error messages and messages that require manual intervention still go to the console.

To reduce the number of key processing messages routed to the console, add a SYSPRINT DD statement to the CAS9 procedure.

The data set defined by SYSPRINT is a standard sequential data set or a JES2 sysout data set. DCB attributes are fixed for 80-byte unblocked records and cannot be overridden via JCL, which means that the DCB must be (LRECL=80,BLKSIZE=80,RECFM=F).

Note: If the SYSPRINT ddname is not suitable, you can nominate another ddname for the flat file through the SYSPDDNM subparameter in the EXEC statement of the CAS9 procedure. For example, the following statement nominates SYSLIST as the ddname and you can reroute CA LMP messages by specifying a SYSLIST DD statement:

```
//CAIRIM EXEC PGM=CAIRIM,PARM='SYSPDDNM(SYSLIST),¼'
```

Optionally Turn Off CAS9115I Messages

CAIRIM issues CAS9115I messages for each RIMPARM statement and each LMP statement read in. You can optionally turn off the CAS9115I messages.

To turn off the RIMPARM CAS9115I messages, add the following statement as the first statement in the RIMPARM member in column 1:

```
ECHO=NO
```

To turn off LMP CAS9115I messages, add the following statement as the first statement in the LMP member in column 1:

```
ECHO=NO
```

If you want to turn on CAS9115I messages from a specific point forward, add the following statement at that point in the LMP or RIMPARM statement member:

```
ECHO=YES
```

Emergency Key Generator

Although CA LMP permits your CA Technologies solutions to run uninterrupted regardless of the CPU on which they are running, enforcement software messages are issued. In emergency situations, such as disaster recovery, you can use the Emergency Key Generator (EKG) to activate your solutions and suppress these messages.

To use EKG, you need an EKG code. Request an EKG code from CA TLC.

The EKG code is good for only ten days after use, based on the Greenwich Mean Time (GMT).

After you have obtained an EKG code, you can activate EKG in one of the following ways:

- Using the Start Command
- Using the KEYS data set member
- Using the PARM keyword

Activate EKG Using the START Command

You can activate EKG using the START command to suppress messages. The START command requires manual intervention.

To activate EKG using the START command

1. Issue the following console command:

```
START CAS9, , ,EKG
```

The CAS9117A message appears and prompts you for an EKG code.

Important! You have three attempts at entering a valid EKG code before EKG processing is aborted.

2. Reply to the message.

On acceptance of the code, EKG is activated.

Example: Console Display

This example shows the console display when running CAIRIM with the EKG parameter:

```

S CAS9, , ,EKGP                               ❶
00 CAS9117A - ENTER EKG DEVICE CODE OR CANCEL TO ABORT ❷
R 00,1A2B3C4D                                  ❸
CAS9116I - EKG DEVICE CODE ACCEPTED ACTIVATED ON:    ❹
    AUGUST 28, 2008 G.M.T
CAS9115I - INPUT: PROD(L0) CPU(3090-600 /011111) DATE(28AUG11)
    LMPCODE(XXXXXXXXXXXXXXXXXX)
CAS9190I - PRODUCT L0 KEY ACCEPTED FOR THIS CPU      ❺
CAS9115I - INPUT: PROD(S0) CPU(3090-600 /099999) DATE(28AUG11)
    LMPCODE(YYYYYYYYYYYYYYYYYY)

```

- ❶ CAIRIM is started with the EKG parameter.
- ❷ The operator is prompted for an EKG code.
- ❸ The operator replies.
- ❹ The EKG code is accepted and activated on August 28, 2008 GMT.
- ❺ The execution key for the indicated product is accepted for the CPU, even if the computer type and serial number do not match what is contained in the execution key.

Activate EKG Using the KEYS Data Set Member

You can activate EKG by including an EKG control statement in the KEYS data set member to suppress messages.

To activate EKG using the KEYS data set member

1. Add the following EKG control statement to the CAW0OPTN(KEYS) member:

Important! The EKG control statement must be the first uncommented control statement in the KEYS member.

```
EKG(nnnnnnnn)
```

```
nnnnnnnn
```

Is the EKG code.

2. Run CAIRIM.

EKG is activated.

Example: Output

This example shows the output when EKG is activated:

```

CAS9075I - SERVICE(CA-RIM/BASE ) VERS(1200) GENLVL(0808AW000)
CAS9115I - INPUT: *
CAS9115I - INPUT: * KEY PARAMETERS FOR LMP
CAS9115I - INPUT: *
CAS9115I - INPUT: EKG(1A2B3C4D) ❶
CAS9116I - EKG DEVICE CODE ACCEPTED ACTIVATED ON: ❷
      AUGUST 28, 2008 G.M.T
CAS9115I - INPUT PROD (L0) CPU(3090-600 /011111) DATE(28AUG11)
      LMPCODE(XXXXXXXXXXXXXXXXXX)
CAS9190I - PRODUCT L0 KEY ACCEPTED FOR THIS CPU ❸
CAS9115I - INPUT PROD (S0) CPU(3090-600 /099999) DATE(28AUG10)
      LMPCODE(YYYYYYYYYYYYYYYYY)

```

- ❶** EKG is the first uncommented control statement, where 1A2B3C4D is the EKG code.
- ❷** The EKG code is accepted and activated on August 28, 2008 GMT.
- ❸** The execution key for the indicated product is accepted for the CPU, even if the computer type and serial number do not match what is contained in the execution key.

Activate EKG Using the PARM Keyword

You can activate EKG by including a PARM keyword in the EXEC statement of the CAS9 procedure to suppress messages. It requires manual intervention.

To activate EKG using the PARM keyword in the EXEC statement

1. Update the EXEC statement in the CAS9 procedure as follows:

```
//CAIRIM EXEC PGM=CAIRIM,PARM=EKGP
```

2. Start the CAS9 procedure.

The CAS9117A message appears and prompts you for an EKG code.

3. Reply to the message.

On acceptance of the code, EKG is activated.

Example: Session Display

This example shows the session when executing CAIRIM with the PARM=EKGP parameter:

```
00 CAS9117A - ENTER EKG DEVICE CODE OR CANCEL TO ABORT      ❶
R 00,1A2B3C4D                                             ❷
CAS9116I - EKG DEVICE CODE ACCEPTED ACTIVATED ON:
    AUGUST 28, 2008 G.M.T
CAS9115I - INPUT: PROD(L0) CPU(3090-600 /011111) DATE(28AUG12)
    LMPCODE(XXXXXXXXXXXXXXXXXX)
CAS9190I - PRODUCT L0 KEY ACCEPTED FOR THIS CPU          ❸
CAS9115I - INPUT: PROD(S0) CPU(3090-600 /099999) DATE(28AUG12)
    LMPCODE(YYYYYYYYYYYYYYYYYY)
```

- ❶ The operator is prompted for an EKG code.
- ❷ The operator replies.
- ❸ The EKG code is accepted and activated on August 28, 2008 GMT.
- ❹ The execution key for the indicated product is accepted for the CPU, even if the computer type and serial number do not match what is contained in the execution key.

System Authorization Facility Support

CAISSF requires no coding or customization when using a CA Technologies security product.

For other security products that support system authorization facility (SAF) (for example, RACF), CAISSF provides a user-modifiable exit, CAS9SAFC. The exit is called if CA ACF2 or CA Top Secret is not installed and if SAF support is active, as determined by the presence of a valid address in the CVTSAF field of the CVT.

A pre-built exit, CAW0LOAD(CAS9SAFC), is installed with the CAIRIM service to support security calls from STC, TSO or batch tasks. The library in which the exit resides must be APF-authorized. The authorization is required to comply with SAF restrictions for signon and resource checking.

If you need to customize the exit, the source code is in the CAW0SRC(CAS9SAFC) member.

Note: If you use a security product that is not from CA Technologies, you must evaluate whether it can handle the anticipated volume of security processing, especially if you customize the CAS9SAFC exit.

CA LMP Errors

If you are using CA LMP, you may encounter error messages during system startup or when you attempt to use a CA LMP controlled product. For information about these messages, see the *Message Reference Guide*.

For assistance, contact CA Total License Care (TLC). For more information, see Contact Technical Support.

LMP Key Check Invocation Exit

If you need to monitor which LMP checks are being made by various CA Technologies products, you might find it useful to use the CAS9FLO0 exit point. The sample exit performs a simple WTO for each LMP key check made and the exit is called for CA Technologies product LMP key checks. The source code to a sample exit is supplied in the CAW0SRC data set.

The CAS9FLO0 LMP exit point module passes one parameter, the address of the two-byte LMP product code that has just been checked, in register 1.

Unless the LMP key checking CA Technologies application runs in key 1-7, CAS9FLO0 is invoked in problem state, key 8.

If the CA Technologies application runs in key 1-7 then the exit is invoked in that key. For example, CA ACF2 runs in key 1 so the LMP exit is called in key 1 for an ACF2 LMP key check. The exit is called using standard linkage conventions.

The LMP exit may be called under a job step task that is authorized. Therefore, you should take the same security precautions regarding access to the installation of this exit as you would take for a z/OS SMF exit.

To turn on the call to CAS9FLO0, add the following statement to the beginning of the RIMPARM statements member and run CAIRIM:

```
LMPEXIT=YES
```

CAS9FLO0 refreshment capability is available through the CAIRIM CAS9INIT REFRESH(LMP) parm function. If CAIRIM is run with this RIMPARM statement, CAS9FLO0 is reloaded into storage along with the other main LMP modules, which allows CAS9FLO0 to be updated without an IPL.:

```
PRODUCT(CAIRIM) VERSION(CAS9) INIT(CAS9INIT) PARM(REFRESH(LMP))
```

LMP CA Product Usage Registration

CA product LMP key checks, by default, cause SMF 89 usage data records to be generated. The SMF 89 record generation is required for customers who have selected site keys for managing or controlling the use of CA software.

If you need to turn this feature off, add the following statement to the CAIRIM RIMPARM member:

```
LMPAM=NO
```

Note: The statement must be placed in column 1.

Setting LMPAM to NO turns off SMF 89 record generation for the life of the IPL. This is to prevent CAS9 from turning the feature on if run for some minor purpose, such as adding a single LMP key. To turn the feature on after running CAS9 with LMPAM=NO, run CAS9 with the following RIMPARM:

```
LMPAM=YES
```

Procedure CASMF89R of the CAI.CAW0PROC data set is used to run the IBM usage data report utility IFAURP. You can copy CASMF89R to one of your system proclibs and customize it for your site. CASMF89R also needs appropriate security definitions assigned.

CASMF89R as delivered, creates a temporary data set of all SMF 89 records from the chosen 'SYS1.MANx' data set then sorts the SMF 89 records and uses the sorted temporary data set as input to the IFAURP utility.

The IFAURP utility creates a report that shows information about CA products that have been used on the system during the timeframe that the chosen 'SYS1.MANx' data set was written to.

The CASMF89R sample proc refers to three parm members. The CAI.CAW0OPTN data set contains members SMF89DMP, SMF89S, and SMF89U. You can copy these members to a parmlib data set of your choice and edit your CASMF89R proc to set the OPTLIB variable default to this parmlib data set. Parmlib member SMF89U requires customization.

You must set your company name and address, your name, phone number, processor type, model, and serial number, and the report data start date. The z/OS command 'D M=CPU' is useful for gathering processor information.

To see which 'SYS1.MANx' data set is currently active, issue a z/OS 'D SMF' console command. You can then run CASMF89R with the following command:

```
S CASMF89R,MAN=x
```

where x is the active 'SYS1.MANx' data set shown in the D SMF display.

You might want to customize a batch job for running the IBM IFAURP utility in order to produce a usage report for a timeframe that makes sense for your site. The following sample JCL steps might be helpful in setting up such a job.

SMF 89 record offloading sample

```
//XXXXXXXX JOB .....
//*
//DMPSMF EXEC PGM=IFASMPDP
//SYSPRINT DD SYSOUT=*
//DUMPIN DD DSN=SYS1.MAN?,DISP=SHR <=== Set MAN data set
//DUMPOUT DD DSN=???.SMFTYP89.RECORDS,DISP=(NEW,CATLG), <=== SET HLQ
// SPACE=(TRK,(100,50),RLSE),DCB=(RECFM=VBS,BLKSIZE=4096),
// UNIT=SYSDA,VOL=SER=vvvvvv <=== Set volser
//SYSIN DD *
        INDD(DUMPIN,OPTIONS(DUMP))
        OUTDD(DUMPOUT,TYPE(089:089))
/*
```

SMF 89 record sorting/filtering sample

```
//XXXXXXXX JOB .....
//*
//IFAUSORT EXEC PGM=SORT
//SYSOUT DD SYSOUT=*
//SORTIN DD DISP=SHR,
//          DCB=BUFNO=16,
//          DSN=HLQ.SMF.RECORDS <=== Offloaded SMF records all types
//*
//SORTOUT DD DISP=(NEW,CATLG),
//          UNIT=SYSDA,
//          SPACE=(CYL,(3,20),RLSE),DCB=*.SORTIN,
//          DSN=HLQ.SMFTYP89.RECORDS <=== Only SMF 89 records output
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SORTDIAG DD DUMMY
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,(1),RLSE)
//SORTWK02 DD UNIT=SYSDA,SPACE=(CYL,(1),RLSE)
//SORTWK03 DD UNIT=SYSDA,SPACE=(CYL,(1),RLSE)
//SYSIN DD *
        OPTION VLSHRT
        SORT FIELDS=(5,250,CH,A)
        INCLUDE COND=(6,1,BI,EQ,X'59')
/*
```

CA Product usage data report sample-

```
//XXXXXXXX JOB .....
/*
```

```
//IFAUSAGE EXEC PGM=IFAURP
//STEPLIB DD DSN=SYS1.SIFALIB,DISP=SHR
//SMFDATA DD DISP=SHR,DSN=HLQ.SMFTYP89.RECORDS <== SMF 89 recs dsn
//SYSUDUMP DD SYSOUT=*
//SYSMSGGS DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUSAGE DD SYSOUT=*
//SYSHIN DD DUMMY
//SYSHOUT DD DUMMY
//SYSIN DD *
*
* CUSTOMER NAME AND ADDRESS
*
  CUSTOMER(
    NAME('Your Company Name')
    ADDRESS(
      'Company addr line 1'
      , 'Town, ST ZIPXX'
    )
    CONTACT('Your name')
    PHONE('(000) 000-0000')
  )
*
* VENDOR NAME AND ADDRESS
*
  VENDOR(
    PRODOWNER('CA')
    NAME('CA')
    ADDRESS(
      'CA INC.'
      , '1 CA PLAZA'
      , 'ISLANDIA, NY 11749'
      , 'FAX: 1-800-000-1212 ATTN: USAGE PRICING'
    )
    CODE(0000000)
  )
*
* PROCESSOR CONTROL STATEMENT
*
  PROCESSOR( (tttt,mmm,sssss)
    PRODUCT(
      PRODOWNER('CA')
      PRODNAME('CAIFLOID')
      START(yyymmdd)
    )
  )
/*
```

where *tttt* = Processor type (such as 2094), *mmm* = Model (such as 712), *sssss* = Serial number, and *yyyymmdd* = Report start date year, month, and day (such as 20070921 for September 21, 2007).

CAISSF

CAISSF (CA International Standard Security Facility) is a subcomponent of the CAIRIM service that provides an external security mechanism for controlling and monitoring access to all CA-defined resources.

CAISSF is integrated into many CA enterprise solutions and also it is used by other CA Common Services for z/OS services. CAISSF provides security services for user sign-in, resource access control, process usage control, and recording and monitoring violation activity. These security services work in conjunction with any z/OS external security product, including CA ACF2, CA Top Secret, and IBM RACF.

Note: For information about the features and functions of the security interfaces to CA ACF2 and CA Top Secret for a CA Technologies product, see the documentation for that product.

Reporting Licensed Registered Product Usage

Product usage is tracked for LMP key licensed or maximum concurrent seat license usage CA products. CA products that employ product usage monitoring, invoke a CA common service at strategic points in time to record usage. The CA product usage common service uses the IBM z/OS Product Registration component to perform the tracking, recording and reporting. The registered product information is stored as z/OS SMF (System Management Facility) type 89 subtype 2 records. z/OS generates these particular SMF record types on an SMF interval basis as defined by the installation.

The SMF type 89 records are collected over the required reporting period. A registered software usage report can be produced from the SMF type 89 records. The usage report is generated using the IBM z/OS product usage report utility program, IFAURP. IFAURP supports specification of software vendor which allows reporting of CA products only.

Follow these steps to produce licensed registered usage software reports:

1. Specify that the z/OS SMF component is to record SMF type 89 records. Make this specification on each system running CA licensed registered usage software programs.
2. Record SMF type 89, subtype 2 records in SMF installation, archival, and retrieval procedures.
3. [Generate the Software Product Registration Report](#) (see page 127) to report product maximum concurrent registered usage.

Record SMF Type 89, Subtype 2 Records

Specify in the logical PARMLIB SMFPRMxx member, the records that SMF is to record. The Software Product Registration Report produced by the IFAURP report utility requires the SMF type 89, subtype 2, records as input. Select SMF type 89 in the SMFPRMxx member, or minimally, select the SMF type 89, subtype 2 records. For information describing the recording of SMF type 89 records, see the IBM publication, *z/OS MVS System Management Facilities (SMF)*.

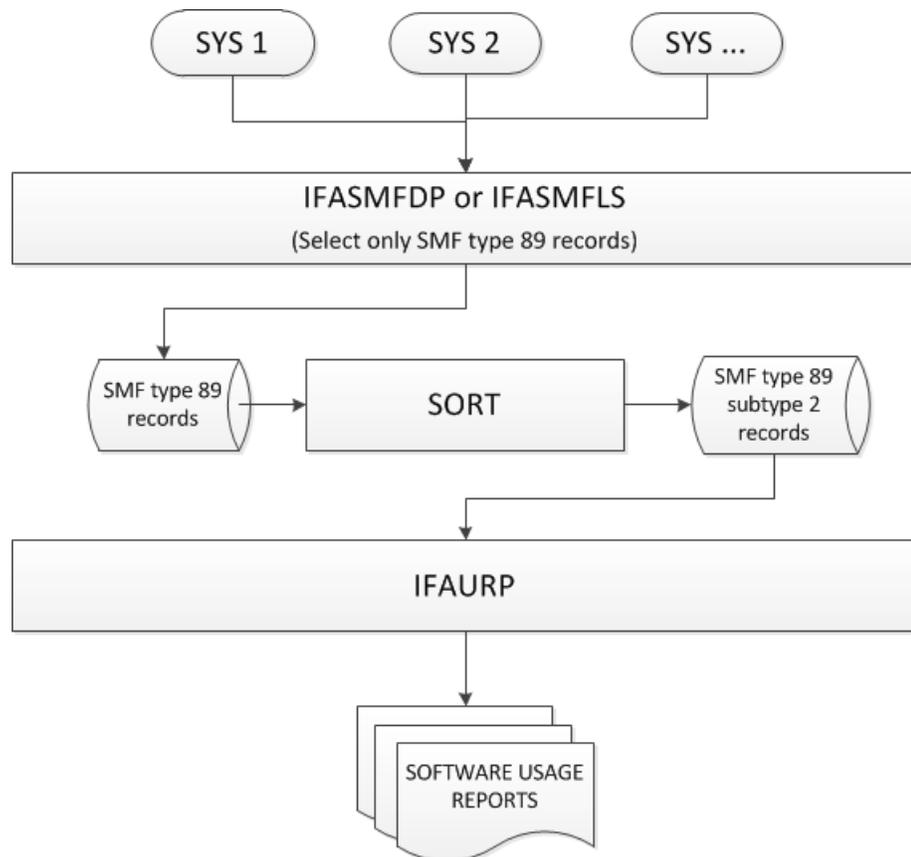
Typically an installation has existing procedures and methodologies for collecting and archiving SMF records for later retrieval and reporting. Verify that the SMF type 89, subtype 2, records are included in the installation managed SMF historical data sets. Record these records for at least as long as production of the Software Product Registration Report requires.

Generate the Software Product Registration Report

The IBM product registration reporting utility, IFAURP, is used to analyze and summarize the SMF type 89, subtype 2, records. These records are used to produce the Software Product Registration Report. This report details the product maximum concurrent registration for each CA product and validates seat license usage compliance. For information regarding use of the IFAURP report utility, see the IBM publication, *z/OS MVS Product Management*.

When producing the Software Product Registration Report, supply the SMF type 89, subtype 2 records from all required systems for the desired reporting period. The general-purpose SMF dump utility, IFASMFDP, can be used to preprocess the SMF records. If SMF records are being recorded to a z/OS log stream, IFASMFLS is used instead of IFASMFDP. Once the target SMF records have been selected, they are sorted before presenting them to the IFAURP utility.

The following diagram illustrates the data flow required to produce reports through the IFAURP report utility:



Sample JCL for Generating the Software Product Registration Report

This sample code illustrates the following:


```

INDD(INDD1,OPTIONS(DUMP))
INDD(INDD2,OPTIONS(DUMP))
INDD(INDD3,OPTIONS(DUMP))
OUTDD(OUTDD1,TYPE(89))
DATE(yyddd,yyddd)
/*
/**
/*******
/**
/**          Sort the SMF type 89 records into a temporary data set      *
/**
/*******
/**
//SORT89 EXEC PGM=SORT,REGION=6M
//SYSOUT DD SYSOUT=*
//SORTIN DD DSN=&&DUMP89,DISP=(OLD,DELETE)
//SORTOUT DD DSN=&&SORT89,DISP=(NEW,PASS),
// UNIT=SYSALLDA,SPACE=(CYL,(5,10),RLSE),
// DCB=*.SORTIN
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SORTDIAG DD DUMMY
//SORTWK01 DD UNIT=SYSALLDA,SPACE=(CYL,(10),RLSE)
//SORTWK02 DD UNIT=SYSALLDA,SPACE=(CYL,(10),RLSE)
//SORTWK03 DD UNIT=SYSALLDA,SPACE=(CYL,(10),RLSE)
//SYSIN DD *
        OPTION VLSHRT
        SORT FIELDS=(5,250,CH,A)
        INCLUDE COND=(6,1,BI,EQ,X'59')
/*
/**
/*******
/**
/**          Generate Software Usage Reports                                *
/**
/*******
/**
//IFAUSAGE EXEC PGM=IFAURP,REGION=4M,
// PARM='PRODUCT'
//STEPLIB DD DSN=SYS1.SIFALIB,DISP=SHR
//SMFDATA DD DSN=&&SORT89,DISP=(SHR,PASS)
//SYSUDUMP DD SYSOUT=*
//SYSUSAGE DD SYSOUT=*
//SYSMSGs DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSHIN DD DUMMY DSN=<input history file spec>
//SYSHOUT DD DUMMY DSN=<output history file spec>
//SYSIN DD *
*

```

```
* Customer name and address
*
CUSTOMER(
    NAME('B & K Enterprises, Inc')
    ADDRESS(
        '1313 Mockingbird Lane'
        , 'Mockingbird Heights, CA 90210'
    )
    CONTACT('M J Borghi')
    PHONE('(555) 555-5555')
)

*
* Vendor name, address, and installation's customer number
*
VENDOR(
    PRODOWNER('CA')
    NAME('CA Technologies')
    ADDRESS(
        'One CA Place'
        , 'Islandia, NY 11749'
        , 'Fax: 1-631-342-6800 ATTN: Usage Pricing'
    )
    CODE(1234567)
)

*
/*
//
```

CAS9INIT CAIRIM Initialization Routine

The CAS9INIT initialization routine lets you control various functions of CAISSF and CA LMP dynamically with the following advantages:

- Ensures that the latest version of CAISSF is loaded and executed
- Shares one set of CAISSF routines across all address spaces
- Re-initializes CAISSF routines through the execution of CAIRIM if maintenance is applied
- Deletes CAISSF routines through the execution of CAIRIM (if desired)
- Refreshes CA LMP
- Refreshes the CAIMB838 SMF intercept
- Refreshes the CA Health Checker common service infrastructure model
- Refreshes the Serviceability subcomponent in common storage

Important! CAIRIM control statements cannot go beyond column 72.

CAS9INIT Parameter Statement

To invoke CAS9INIT functions, you add the following parameter statement in the CAIRIM parmlib member, CAWOOPTN(CARIMPRM):

```
PRODUCT(CAIRIM) VERSION (CAS9) INIT (CAS9INIT) PARM(keyword(value))
```

The PARM parameter has the following keywords:

REFRESH

Refreshes the indicated components (for example, after maintenance):

- LMP indicates CA LMP.
- MB838 indicates the CAIMB838 SMF intercept.
- SSF indicates CAISSF.
- HCHECK indicates CA Health Checker common service.
- SERVABIL indicates the Serviceability subcomponent.

The keyword can have multiple values.

Example: The following statement refreshes all three components.

```
PRODUCT(CAIRIM) VERSION(CAS9) INIT(CAS9INIT) -  
PARM(REFRESH(SSF,MB838,LMP))
```

Example: The following statement refreshes CAISSF.

```
PRODUCT(CAIRIM) VERSION(CAS9) INIT(CAS9INIT) PARM(REFRESH(SSF))
```

Example: The following statement refreshes CA Health Checker.

```
PRODUCT(CAIRIM) VERSION(CAS9) INIT(CAS9INIT) PARM(REFRESH(HCHECK))
```

Example: The following statement refreshes Serviceability.

```
PRODUCT(CAIRIM) VERSION(CAS9) INIT(CAS9INIT) PARM(REFRESH(SERVABIL))
```

SSF

Lets you control CAISSF as follows:

- ACF2 enables CAISSF to initialize before CA ACF2.
- TSS enables CAISSF to initialize before CA Top Secret.
- RACF enables CAISSF to initialize before RACF.
- DELETE removes CAISSF.
- NONE
- REINIT re-initializes CAISSF.

Example: The following statement enables CAISSF to initialize before CA Top Secret.

```
PRODUCT(CAIRIM) VERSION(CAS9) INIT(CAS9INIT) PARM(SSF(TSS))
```

If you do not specify PARM in the statement, CAS9INIT will initialize CAISSF.

How CAS9INIT Initializes CAISSF When Security Product Is Not Yet Active

If CAS9INIT finds no active security product (for example, when CAIRIM is run before the security product becomes active), the following messages are issued to the system console, prompting the operator to identify the security product:

```
CAS9075I - SERVICE(CA-RIM/BASE ) VERS(1200) GENLVL(0808AW000)
CAS9115I - INPUT: *
CAS9115I - INPUT: PRODUCT(CAIRIM) VERSION(CAS9) INIT(CAS9INIT)
CAS9025A - NO SECURITY SYSTEM AVAILABLE
00 CAS9026A - REPLY WITH SECURITY SYSTEM,"ACF2","TSS","RACF" OR
              "N" TO CANCEL
```

If the reply identifies the security product, CAS9INIT continues to initialize CAISSF.

If the reply is N, the following message is issued and CAS9INIT fails to initialize CAISSF:

```
CAS9021E ENVIRONMENT ERROR DETECTED. UNABLE TO ADD CAISSF ROUTINES
```

You can bypass the operator prompt by adding the following parameter statement in the CAIRIM parmlib member, CAWOOPTN(CAIRIMPRM):

```
PRODUCT(CAIRIM) VERSION(CAS9) INIT(CAS9INIT) PARM(SSF(value))
```

value

Indicates one of ACF2, TSS, or RACF.

CAISSF RACF Class Table Parameters

CAISSF needs a RACF Class table to identify how security calls are processed. A default table is created, but certain CA products require additional entries. These entries are described in the documentation for the related product. The control statements for this table are read from a CAIRACF DD statement if it is present in the CAS9 procedure. The following is the format of each of the statements:

```
RACFCLASS CA-solution classname,translated classname,
          FASTAUTH=NO|YES,CICS=NO|YES
```

- Any statement with an asterisk (*) in column 1 is ignored.
- The operation is RACFCLASS and must be followed by at least one blank.
- The *CA-solution classname* is required and is the value of the class name used by the CA application.
- The *translated classname* is required and is the value used by RACF to verify proper authorization for access to the given class.

- FASTAUTH=YES means that the Fast RACHECK (FRACHECK) is used to authenticate access to the given class. The default is FASTAUTH=NO
- CICS=YES means that the Fast RACHECK is used with a CICS application. CICS=YES implies FASTAUTH=YES. The default is CICS=NO which means this class entry is not used under CICS.

CAMODID

The CAMODID is a CA Technologies z/OS TSO command processor utility which finds the #MODID blocks in load modules located in common storage or selected load module datasets.

The #MODID block contains the product-related information including the service level of the load module element.

The output includes a summary of service maintenance installed based on the located #MODID blocks.

The CAMODID TSO Command Utility is available with the following releases:

- CA Common Services R14.0 PTF RO44913
- CA Common Services R14.1

Using CAMODID

Use the following procedure to begin using the CAMODID utility.

Follow these steps:

1. Install CA Common Services maintenance and IPL the system.
2. CAMASTER address space at startup builds the infra-structure.
3. Log on to TSO.
4. Issue 'CAMODID HELP' from ready mode or ISPF option 6.
5. Review output from HELP including functions, parameters and examples.

The CAMODID utility provides the following features:

- The service summary is sorted by RMID(PTF).
- Allows masking for LMOD, FMID and RMID parameters.
- Can be run in BATCH (IKJEFT01) for hardcopy or longer running functions.
- Sample CAM REXX can be used to capture output to a temp file for online browsing.

The CAMODID utility has the following limitations:

- Requires z/OS 1.10 and above.
- Output is limited to modules that contain #MODID blocks.

Functions (CAMODID)

HELP

Provides command syntax, operands and examples.

SYSTEM

Scans modules in common storage and current linklist datasets for #MODID blocks.
Common includes nucleus, LPA and MLPA .

DSNAME

Scans target dataset for #MODID blocks.

DDNAME

Retrieves the datasets of the target DDNAME and reports on #MODID blocks in conjunction with the JOBNAME parameter.
Useful for identifying STEPLIB concatenations or overrides.

Operands (CAMODID)

DETAIL

Provides detail information from #MODID blocks.

FMID

Limits search to matching fmid(s). Value can be masked.

JOBNAME

Targets specific address space when used in conjunction with DDNAME function. If omitted the current job will be used.

LMOD

Limits scan to matching load module(s). Value can be masked.

NOCOMMON

Excludes load modules from common storage for SYSTEM function.

NOLNKLST

Excludes load modules from linklist datasets for SYSTEM function.

PTF

Limits scan to matching ptf(s). Value can be masked.

Invoke CAMODID

To invoke CAMODID and view output in a temporary dataset, use the following sample CAM REXX EXEC :

```
/* REXX */
/* WRITE CAMODID OUTPUT TO TEMP DATASET FOR EASE OF VIEWING */
PARSE ARG P1 P2 P3 P4 P5 P6 P7 P8 P9
USER = USERID()
SYS = MVSVAR('SYSNAME')
UDSN = USER'.CAMODID.'SYS'.TEMPDSN'
IF SYSDSN('"'UDSN"'") = OK THEN
  "ALLOC F(MODDSN) DA('"'UDSN"'') SHR REU"
ELSE
  "ALLOC F(MODDSN) DA('"'UDSN"'') NEW SP(25 15)
  TR DSORG(PS) RECFM(F B) LRECL(80) BLKSIZE(3120)"
X = OUTTRAP('MOD.')
```

```
TRACE(OFF)
"CAMODID "P1" "P2" "P3" "P4" "P5" "P6" "P7" "P8" "P9"
"EXECIO * DISKW MODDSN (STEM MOD. FINIS"
"BROWSE "'UDSN"'")
"FREE F(MODDSN)"
EXIT(0)
```

JCL to run CAMODID

To run CAMODID using batch TMP, use the following sample JCL and modify to suit your requirements:

```
//USERIDA JOB (1), 'OS/MVS', MSGCLASS=X, MSGLEVEL=(1,1),
//          NOTIFY=USERID
//STEP1 EXEC PGM=IKJEFT01
//SYSPRINT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//SYSTSIN DD *
CAMODID SYSTEM
//
```

Chapter 5: Event Notification Facility

This section contains the following topics:

[Overview](#) (see page 137)

[Features](#) (see page 138)

[Operation](#) (see page 138)

[Control Options Sharing](#) (see page 146)

[Archive Events](#) (see page 147)

[Restore Events](#) (see page 149)

[CAIENF/CICS Operation](#) (see page 150)

[CAIENF/CICS SPAWN Communications Facility Operation](#) (see page 154)

[CAIENF/DB2 Operation](#) (see page 155)

[CAIENF/USS Operation](#) (see page 157)

[SNMP Monitor](#) (see page 158)

Overview

The Event Notification Facility (CAIENF) is an operating system interface service that enables CA Technologies applications to obtain event data from z/OS and subsystems such as CICS, DB2, and USS. The centralization of operating system interfaces in CAIENF provides important efficiency benefits while enhancing overall system integrity and providing a single point of control.

CAIENF includes the following subcomponents:

- CAIENF/CICS
- CAIENF/CICS/SPAWN
- CAIENF/DB2
- CAIENF/USS
- SNMP Monitor

Features

CAIENF provides the following features:

- **Single Operating System Interface**—A single interface lets CA Technologies applications minimize dependencies on IBM-provided exits and intercept points. As a result, new operating system releases do not affect applications that use CAIENF.
- **Ease of Customization**—Many aspects of CAIENF operation are specified by control options that are processed at startup. You can override most control options at any time by issuing a CAIENF command from any operator console. Changes to most control options have no impact on running applications.
- **Dynamic Installation and Reconfiguration**—All operating system intercepts are installed at CAIENF startup without needing any IPLs or permanent modifications to IBM or vendor code. You can refresh critical CAIENF modules dynamically to apply (or remove) emergency maintenance without impacting the applications that use CAIENF services.
- **High Performance**—CAIENF exploits the latest in operating system and hardware technology to ensure that the overall performance, reliability, and serviceability of CA Technologies applications meet the needs of the most demanding installations.
- **Relational Database Technology**—Relational database technology is exploited for product data as well as for logging and recovery. The database provides enhanced recoverability, enabling products to retrieve data they may have missed while shutting down or because of application failure. The control options in CAIENF let you manage z/OS logging and recovery data for all products in a simple, consistent manner.

The database provides a central mechanism for reporting and auditing; many applications can share the same database.

- **Built-in Diagnostic Aids**—Diagnostic aids simplify problem determination and reapplication tasks. Messages, traces, and dumps are produced when an error occurs, and can be requested manually.

Operation

The basis for CAIENF processing is the event—a logical condition that needs to be broadcast to other products. Events are initiated by the following conditions:

- Built-in CAIENF intercepts, including batch jobs starting or ending, or files being accessed.
- Other CA Technologies products, such as triggering of SNMP traps in response to internal product-defined conditions.

CA Technologies products use a formal set of CAIENF programming interfaces to process events and to access event data.

View Event Names

To view the event names in the CAIENF database, issue the following command from a system console:

```
ENF EVENT
```

Run CAIENF as a Started Task

The CAIENF service runs as a started task in its own address space. The user ID of the started task must have a valid OMVS segment.

Important! CAIRIM prepares the environment for your CA Technologies products. You must not start CAIENF before CAIRIM ends, when the environment is ready.

To run CAIENF, issue the following command:

```
START ENF
```

The CAIENF service is started when the following message appears:

```
CAS9200I  CA-EVENT NOTIFICATION FACILITY ACTIVE
```

To ensure that CAIENF is started in a timely fashion after each system IPL, you can define the command as a z/OS auto command in CAIRIM.

Run CAIENF under the Master Subsystem

Some CA Technologies products require initialization before JES and require CAIENF to be started with SUB=MSTR.

Note: Running CAIENF with SUB=MSTR reduces diagnostic capability. You should not run CAIENF with SUB=MSTR unless there is a substantial benefit from doing so. As an alternative, you can use CAIENF auto commands. For example, to guarantee that CAIENF initializes before the beginning of batch job activity, you can define JES initiators as drained. Then, in the CAIENF auto commands member, you can use a \$SI command to start the desired initiators.

To run CAIENF under the Master subsystem

1. Ensure that a copy of the CAWOPROC(ENF) member, updated to suit the requirements of your site, is in a system proclib and not in a proclib defined to JES. Review the following DD statements:
 - When running CAIENF under the Master subsystem, you cannot direct SYSPRINT to SYSOUT=*. You must specify a SYSPRINT DD DUMMY statement or, if you want the SYSPRINT output, direct SYSPRINT to a permanent data set.
 - If CAICCI is installed, a TRCPRINT DD statement is required. It has the same restrictions as SYSPRINT.
2. Ensure that all data sets accessed using the ENF procedure are cataloged in the master catalog.
3. Use the START command to start ENF before JES at the next IPL. For example:

```
START ENF, SUB=MSTER
```

Stop CAIENF

Important! You need to exercise caution when stopping CAIENF. Products that rely on the CAIENF service cannot function correctly until CAIENF is restarted.

To stop CAIENF, issue the following command:

```
STOP ENF
```

Restart CAIENF

If CAIENF is terminated due to an operational or system error, you can restart it.

To restart CAIENF, issue the START ENF command.

CAIENF detects that it is being restarted and continues processing using the already loaded modules and control blocks.

CAIENF Start Options

The following CAIENF control options can be issued only through the START ENF command:

AUTOCMDS

Processes all CAIENF auto commands.

The CAIENF auto commands member is processed once for each IPL. If you want the member to be reprocessed when CAIENF is stopped and restarted, you must specify AUTOCMDS in the START command as follows:

```
S ENF , , , AUTOCMDS
```

REINIT

Refreshes all of the CAIENF global routines.

Important! Use this option only under the direction of CA Support personnel.

```
S ENF , , , REINIT
```

Example: START Command with Multiple Options

You can specify multiple CAIENF options in a START command by enclosing comma separated options in parentheses:

```
S ENF , , , (AUTOCMDS, REINIT)
```

CAIENF Auto Commands Member

After the successful initialization of CAIENF, CAIENF reads the data set member defined by the ENFCMDS DD statement and issues each command in this member.

The CAIENF auto commands member specifies one command per line, exactly as it would be issued from the operator console. You can add comments by placing an asterisk (*) in Column 1.

Note: The auto commands member is processed once for each IPL. If CAIENF is stopped and restarted, and you want to reprocess the member, you specify the AUTOCMDS option in the START command.

Configure DCM Statements

You must define DCMs to CAIENF by configuring DCM statements. DCM statements must be placed in the CAIENF started task ENFPARMS data set and are processed during CAIENF startup only.

Note: Because integration of the CA-Universe database system within the CAIENF address space has been eliminated, a DD statement defining the CA-Universe database store (DDName: ENFDB) is no longer needed in the CAIENF started task JCL. Through the CA Datacom/AD API, CAIENF communicates to the database service provider through an interface residing in its own address space.

DCM configuration statements follow the same syntax rules as other CAIENF configuration statement or initialization parameter. A DCM statement has the following format:

```
DCM(object [, DDName])
```

object

Identifies the required DCM object member name.

DDName

(Optional) Identifies the optional DDName of the data set where the DCM object resides. If DDName is omitted, DDName "CAIDCM", which defines the default DCM object library, must be present in the CAIENF started task JCL. This library is used to locate DCM objects. DDname may also be specified as an asterisk (*), which indicates that the DCM object resides in the CAIENF started task JOBLIB/STEPLIB concatenation or the LPALST/LNKLST concatenation.

The DDName parameter allows loading of DCMs from CA Technologies product target libraries without including them in the CAIENF started task JOBLIB/STEPLIB and without requiring that the CA Technologies product target libraries are APF authorized.

The following are DCM configuration statement examples:

```
DCM(CAS9DCM2)
```

```
DCM(CAS9DCM4)
```

```
DCM(CARRDCM0, USSLIB)
```

```
DCM(SYSVDCM, *)
```

In these examples, CAS9DCM2 and CAS9DCM4 are loaded from the data set defined by DDName "CAIDCM" (default). CARRDCM0 is loaded from the data set defined by DDName "USSLIB". SYSVDCM is loaded from the CAIENF started task JCL STEPLIB/JOBLIB concatenation or, if that load fails, from the z/OS LNKLST/LPALST concatenation.

Replace a DCM

Sometimes a DCM must be replaced. The DCM replacement may include a definition of a new event or a change to the definition of an event. When CAIENF is recording events to the database, the database table associated with the changed event must be deleted. Instructions are provided along with the new DCM PTF having a ++HOLD status.

Important! Before you submit any jobs to replace a DCM, consider the following:

- A CA Datacom/AD Multi-User Facility (MUF) must be running on the system where the JCL associated with the procedure will execute.
- A CA Datacom/AD MUF can execute in its own address space or under the CAIENF address space.
- To customize CA Datacom/AD for CAIENF or resolve CA Datacom/AD for CAIENF customization issues, the CA Datacom/AD MUF must be running with its own address space and CAIENF must be down or started in a separate address space from the CA Datacom/AD MUF with the control option NODB

To utilize the new DCM

1. Reference the new DCM in the CAIENF startup JCL under DD statement CAIDCM.

Important! If you are running CAIENF with control option NODB, Stop and Restart CAIENF to pick up the new module or proceed to the next step to delete the Database definitions for the old DCM.

2. Prepare batch jobs CASQL004 and CADB001 located in the CAW0JCL data set by editing and specifying a job card and the appropriate Datacom execution libraries in the Set JCL statements.

CASQL004 must have the name of the specific table to be deleted in the SQL Drop Table statement.

Optionally, CAS9DCMR utility can be used to generate the DROP TABLE statements. For more information on CAS9DCMR - DCM MAPPER, see the *Reference Guide*.

3. Shutdown CAIENF with the STOP ENF operation command.

Important! If you are running CAIENF with an internal CA Datacom/AD MUF, after CAIENF is down, you must start the CA Datacom/AD MUF in its own address space before submitting the job that deletes the old DCM's database definitions.

4. Submit CASQL004.

If successful you may restart CAIENF. When CAIENF is restarted, it will dynamically create the new DCM's database definitions.

5. If the CASQL004 batch job completes in error, the database has CAIENF table information remaining in its internal cache. Submit batch job CADB001 to ready the database for another Drop Table.
6. After CADB001 completes, resubmit job CASQL004 and restart CAIENF.

Note: For more details, see the CAIENF Batch Database Query and Administration Appendix.

Handling ENF Database Full Conditions

During ENF operations, you may receive the following messages:

```
DATACOME:DB02406I - PXX END - EXTEND FAILURE DUE TO X37 ABEND
DATACOME:DB01702I - DYNAMIC EXTEND OF AREA ENF00700 HAS FAILED
CAS9316E - STEPTERM "Insert " failed - rc(-258) r/s(X'F5F7E2F0F6').
CAS9340I - TABLE FULL - DBID=700 INTERNAL NAME B22 .
*CAS9208E - CA-ENF DB: DB Insert error SQLCODE=-258 SQLSTATE=57S06
CAS9303E - Event XXXXXXXX no longer recorded due to DB error 0008
...
```

Additional similar messages can appear for errors for other events being recorded. New records are not recorded to the database during this time. However, the CA applications "listening" for these events continue to run without issue because they receive event information before any attempt is made to write the events to the database. If these applications shut down, they cannot recover events (checkpoint) from the database during the timeframe when event recording stopped.

If you experience this situation, check that an ENF ARCHIVE is being done on a regular basis. Check the ENFPARM parameter file for the ARCHIVE control option. During the ARCHIVE process, expired events are deleted from the ENF database. Events expire based on the Retention Period (RP) set for the individual events. If a retention period is not set for the individual event, ENF uses the value that is specified on the DETAIL control option in the ENFPARM parameter file.

If the ENF ARCHIVE is successfully running each day, the ENF database may be too small for the number of event records being recorded. If possible, use SELECT statements to filter events to be written. Information on the SELECT control option (and all ENF control options) can be found in the *CA Common Services Reference Guide*.

If you have not experienced problems with previous ARCHIVE processing and if you cannot further filter the number of events being recorded, do *one* of the following:

- [Increase the ENF database data set size.](#) (see page 145)
- [Free space temporarily.](#) (see page 146)

How to Increase the ENF Database Size

Use this procedure to increase the ENF database size. Before you start this procedure, check that the ENF database is full from a CA Datacom perspective. Customize and run the CAW0JCL CADB006 DBUTLTY CXX report job. The CADB006 job is set up to run with the MUF active.

Look at the tables for the fields:

- BLOCKS IN USE
- BLOCKS UNUSED
- TOTAL BLOCKS

If the BLOCKS UNUSED is 0, increase the ENF database because the tables are full.

Follow these steps:

1. Stop CA products that are currently using ENF for recording events.
2. Stop ENF. If your site is running with an external MUF, shut down the MUF.
3. Customize and run the CAW0JCL CADB007 DBUTLTY job to back up the ENF database.
4. Check the SYSOUT to confirm that the backup completed properly.
5. Customize and run the CAW0JCL CADB008 DBUTLTY job to complete the following tasks:
 - Rename the existing ENF database data set.
 - Allocate and initialize a new ENF database data set.
 - Restore the ENF database backup from step 3.
6. Start ENF (and the MUF if running with external MUF).
7. Start CA products using ENF.

Free Database Space Temporarily

To free space in the database temporarily, you can PURGE events until you can increase the size of the database. You can do so dynamically by using the following console command:

```
ENF PURGE(event_name)
```

This command purges all events for the selected event_name, regardless of the event age. We recommend that you select a highly recorded event to gain the most space. Common examples are as follows:

- DSCLOSE
- STEPTERM

Control Options Sharing

Multiple systems can share a single CAIENF control option member. The method uses the CAIENF IF and ENDIF options to specify groupings of different systems and control options.

The IF option lets you test whether a condition (in this case, a group of systems) is true or false.

- If the condition is true, the options following the IF statement are executed. Execution stops when an ENDIF option is encountered.
- If the condition is false, the options following the IF statement are purged and a message is sent to the console.

The next IF statement is tested, until the last ENDIF statement is reached.

Options that follow an IF statement can include the CAICCI protocol statement (PROTOCOL), System Symbolics such as SYSNAME, or other valid options.

Note: For more information about control options, see the *Reference Guide*.

Example: Sharing of Frequently Used Options

```
IF (SYSPLEX='PLEX01' & SYSNAME='HP94' | SYSNAME='HP97')  
  EVENT(JOBTERM, REC)  
ENDIF()
```

```
IF (SYSNAME='HP91' | SYSNAME='HP92' | SYSNAME='HP94')  
  PROTOCOL(TCPIP)  
ENDIF()
```

Archive Events

To prevent the CAIENF database from filling all available disk space, CAIENF database recorded instances of events can be automatically archived and purged. Purging is performed when an event instance exceeds its retention period, archiving is optional. Event is the ENF control option statement used to designate whether to archive expired event instances or only purge them. ARCHIVE is the ENF control option statement used to designate when the event instance purge and archive should occur.

Upon completion of the archive, CAIENF displays event archiving information for each event. More detailed information is available to administrators by executing batch job CASQL005 located in the CAW0JCL data set. The archive includes information useful for performing a restore. The information includes the archive date and time, the events included in the archive, identification of the first and last events archived for each event, the tape or DASD volser for the archive, and the archive data set name.

Examples:

```
EVENT(JOBFAIL,RP=7)
```

where RP is the number of days the specified event will remain in the database.

```
EVENT(JOBTERM,PURGE=N)
```

where PURGE indicates whether the event instances will be purged and not archived (Y) or purged and archived (N).

```
ARCHIVE(hhmm)
```

The purging and optionally archiving process will be performed each day at the specified time; hour and minute.

```
ARCHIVE(NOW)
```

The purging and optionally archiving process will be performed immediately.

```
ARCHIVE(OFF)
```

The purging and optionally archiving process will not be performed.

Archiving can be specified to write to tape or disk. The configured ENF UNIT and SPACE control options determine the type of archive data set. In addition, other ENF control options determine the archive data set name, space, volser, retention period, label, and cataloging of the data set.

The ENF archive data set control options are:

- **DSNAME** - Controls dynamically allocated archive data sets. The data set name specified is used as a prefix (16 characters in length including periods). The default is ENF.ARCHIVE.
- **LABEL** - Specifies standard (SL) or Non-standard (NL) when the archived back up data set is to tape.
- **RETPD** - Specifies the retention period of the archive data sets. CAIENF keeps track of these allocated archive data set within its database.

- **SPACE** - Controls the primary and secondary space allocation for the archive data set.
- **UNIT** - Specifies TAPE or 3390 as the unit type to be used for the allocated archive data set.

Note: For detailed explanation of these ENF control options, see the *Reference Guide*.

Restore Events

The CAS9DUTL batch database utility supports the RESTORE command to restore event data from one or more archive data sets into a database table associated with the event.

You can restore archived events into either a temporary or an active event table. The description of the table in the database that will receive the archived event data must match the description of the table found in the archive data set selected by the restore utility. By default, event data is restored into a temporary table.

When restoring event data into a temporary table:

- If the temporary table does not exist, it is created for you using information found in the first archive data set that is opened that satisfies the restore request. The temporary table name is created by prefixing the event name with CAS9TMP_.
- If the temporary table already exists, it must match the description of the table in the archive data set, or it must be dropped before running the restore job.

Temporary tables are not selected during the archive/purge process. A temporary table remains defined to the database until you drop the table. You can drop a temporary table using JCL found in the CAW0JCL data set member CASQL004 with a SYSIN statement similar to the following:

```
DROP TABLE CAS9ENFS.CAS9TMP_XXXXXXXX;
```

CAS9TMP_ indicates that the table to be dropped is a temporary table and XXXXXXXX should be replaced with the event table name

To restore event data to the active event table, the JCL to run the restore utility must contain the TBUPDATE parameter on the EXEC card as follows

```
//STEP1 EXEC PGM=CAS9DUTL, PARM='TBUPDATE'
```

When restoring event data into an active event table, the table must already exist in the database. Active event tables are created at CAIENF initialization based upon the EVENT commands specified. The restored event data remains until the next archive/purge process occurs. The newly restored event data is then re-archived and/or purged.

Use the batch job CASQL005 located in the CAW0JCL data set to display information about the archives. Only archives found in the table can be used to restore an event.

The archive table includes the following information:

- **Archive Date** - The day the archive was performed.
- **Archive Time** - Time of day the archive was performed.
- **Event Name** -The name of the archived event.
- **First Date** - The date of the first record in the archive for the event.

- **Last Date** - The date of the last record in the archive for the event.
- **Volser** - The volser of the tape or DASD for the archive.
- **Data Set Name** - The archive data set name.

Use the output information generated by the CASQL005 job to create the appropriate CAS9DUTL restore request.

The EVENT parameter identifies the event and the FROM and TO parameters are used to specify a date range of events to be restored.

Example :

```
RESTORE EVENT(event_name) FROM(yyyyddd) TO(yyyyddd)
```

The batch JCL to execute the RESTORE operation is located in the CAWOJCL data set member CADB002.

CAIENF/CICS Operation

CAIENF/CICS provides an interface into CICS that CA Technologies products can use.

After the CAIENF service starts, it starts the CASCCICS subtask to activate the CAIENF/CICS service.

Messages appear to indicate that CAIENF/CICS is active.

The CAIENF/CICS service installs intercepts in CICS regions to provide an interface for CA products into those regions. The installation of intercepts in CICS regions can be done automatically or manually, using the CAIENF/CICS MODE control option.

Note: To use CAIENF/CICS, its SMP/E function modification identifier (FMID) and CAS9DCM2 database DCM must be installed.

Note: For information about CAIENF/CICS requirements, see the *Installation Guide*.

How CAIENF/CICS Checks for Intercept Modules

The installation of the CAIENF/CICS intercept in a CICS region requires the CAS9Cxx module where xx identifies the CICS release. The modules are distributed in the CAW0LOAD data set.

CAIENF/CICS checks for the module in the following sequence. This is important in case you want to override an intercept module with another intercept module (when testing maintenance). For the intercepts to install, you must ensure that the required modules are in *one* of the indicated locations.

1. CAIENF/CICS searches for the module in the library specified by a CENFLIB DD statement in the CICS startup JCL member. If the statement exists and the module is found, it loads the module into the private address space of the CICS region.
2. If the CENFLIB DD statement does not exist or the module is not found, CAIENF/CICS searches for the module in the STEPLIB of the CICS startup JCL member. If the module is found, it loads the module into the private address space of the CICS region.
3. If the module is not found in the STEPLIB, CAIENF/CICS checks the common storage (CSA). You can load modules into CSA by using the CAIENF/CICS CICSREL control option.

Note: For information about the CAIENF/CICS CICSREL control option, see the *Reference Guide*.

Configure CAIENF/CICS to Install Intercepts Automatically

The CAIENF/CICS interface can be configured to automatically install intercepts in the CICS regions.

To configure CAIENF/CICS to install intercepts automatically, update the CAIENF control option member with the following CAIENF/CICS control option:

```
MODE(CICS,ON)
```

The intercepts will be installed automatically in the appropriate CICS regions when the following occurs:

- CAIENF is initialized after an IPL.
- A CICS region becomes active after CAIENF has been initialized.

Configure CAIENF/CICS to Install Intercepts Automatically in Specific CICS Regions

Depending on your needs, you can configure your environment to install CAIENF/CICS intercepts in specific CICS regions.

To configure CAIENF/CICS to install intercepts automatically in specific CICS regions

1. Make the CAW0LOAD data set available to each CICS region where you want the CAIENF/CICS intercepts installed by one of the following methods:
 - Add a CENFLIB DD Statement, which points to the CAW0LOAD Data set, to the CICS startup JCL member.
Important! If you use CENFLIB, the CAW0LOAD data set which contains module CAILPAM must be in the link list or CICS Steplib.
 - Add the CAW0LOAD data set to the CICS STEPLIB.
2. Update the CAIENF control option member with the following CAIENF/CICS control option:

```
MODE(CICS,ON)
```

Important! You must not use CICSREL(*nn*). Using it would install the intercept for a given CICS release for *all* CICS regions.

When CAIENF is initialized, the intercepts will be installed automatically in the configured CICS regions.

Configure CAIENF/CICS to Install Intercepts Automatically for Specific CICS Releases

Depending on your needs, you can configure your environment to install CAIENF/CICS intercepts in CICS regions that are at specific releases.

To configure CAIENF/CICS to install intercepts automatically for specific CICS releases, update the CAIENF control option member with the following CAIENF/CICS control options:

```
MODE(CICS,ON)  
CICSREL(nn,...)
```

nn

Specifies the CICS release.

CICSREL loads the intercept modules for the specified CICS releases into CSA. To minimize CSA utilization, you should specify only the releases that you actually use.

Note: For more information about the CAIENF/CICS control options, see the *Reference Guide*.

When CAIENF is initialized, the intercepts will be installed automatically in CICS regions that are at the specified releases.

Install CAIENF/CICS Intercepts Manually

Note: You cannot install CAIENF/CICS intercepts for CA ACF2 and CA Top Secret manually in CICS TS regions.

If CAIENF uses the CAIENF/CICS MODE(CICS,NONE) control option, you must install the CAIENF/CICS intercepts manually. Manual installation lets you install the intercepts in CICS regions selectively.

To install the CAIENF/CICS intercept manually in a CICS region, issue the following command:

```
ENF CICS(START,jobname,prodcode)
```

Note: For information about the CAIENF/CICS control option, see the *Reference Guide*.

Activation of CA Technologies Products (CAIENF/CICS)

CA Technologies products that use CAIENF/CICS are defined by the DCM configuration statements in the CAIENF ENFPARMS data set. During initialization, each product defined as a user of CAIENF/CICS can determine whether it needs to process in the current CICS region.

Example: Activate a CA Product in a CICS Region Manually

If CAIENF is configured with MODE(CICS,NONE) and you want to activate CA JARS (product code UX62) in a CICS region named CICS310, issue the following command:

```
ENF CICS(START,CICS310,UX62)
```

Note: You cannot activate CA ACF2 and CA Top Secret manually in CICS CTS regions.

CAIENF/CICS SPAWN Communications Facility Operation

CAIENF/CICS SPAWN is a communications facility that enables CA Technologies products to start units of CICS work from outside the CICS region using CAICCI.

When the CAIENF service starts, it activates the CAIENF/CICS SPAWN service.

The CAIENF/CICS SPAWN service installs intercepts in CICS regions. The installation of intercepts in CICS regions can be done automatically or manually using the CAIENF/CICS SPAWN MODE control option.

Note: To use CAIENF/CICS SPAWN, its SMP/E FMID must be installed.

Note: For information about CAIENF/CICS SPAWN requirements, see the *Installation Guide*.

How CAIENF/CICS SPAWN Checks for Intercept Modules

The installation of the CAIENF/CICS SPAWN intercept in a CICS region requires the CAS9SPxx module where xx identifies the CICS release. The modules are distributed in the CAW0LOAD data set.

CAIENF/CICS SPAWN checks for the modules in the following sequence. This is important to remember in case you want to override an intercept module with another intercept module (when testing maintenance). For the intercepts to install, you must ensure that the required modules are in one of the indicated locations.

1. CAIENF/CICS SPAWN searches for the module in the library specified by a CENFLIB DD statement in the CICS startup JCL member. If the statement exists and the module is found, it loads the module into the private address space of the CICS region.
2. If the CENFLIB DD statement does not exist or the module is not found, CAIENF/CICS SPAWN searches for the module in the STEPLIB of the CICS startup JCL member. If the module is found, it loads the module into the private address space of the CICS region.
3. If the module is not found in the STEPLIB, CAIENF/CICS SPAWN checks the CSA. You can load modules into CSA by using the CAIENF/CICS SPAWN CICSPAWN control option.

Note: For information about the CAIENF/CICS SPAWN CICSPAWN control option, see the *Reference Guide*.

Configure CAIENF/CICS SPAWN to Install Intercepts Automatically

To configure CAIENF/CICS SPAWN to install intercepts in CICS regions automatically, update the CAIENF control option member with the following CAIENF/CICS SPAWN control option:

```
MODE(CICSPAWN,ON)
```

When CAIENF is initialized, the intercepts will be installed automatically in CICS regions.

Install CAIENF/CICS SPAWN Intercepts Manually

If CAIENF uses the CAIENF/CICS SPAWN MODE(CICSPAWN,NONE) control option, you must install the CAIENF/CICS SPAWN intercepts manually.

To install the CAIENF/CICS SPAWN intercept in a CICS region, issue the following command:

```
ENF CICS(START,cics_jobname,SPWN)
```

Example:

To install the CAIENF/CICS SPAWN intercept in a CICS region named CICSPROD, issue the following command:

```
ENF CICS(START,CICSPROD,SPWN)
```

CAIENF/DB2 Operation

The CAIENF/DB2 service installs intercepts in DB2 regions to provide an interface for CA products into those regions.

When the CAIENF service starts, it activates the CAIENF/DB2 service.

The CAIENF/DB2 service installs intercepts in DB2 subsystems using the CAIENF/DB2 MODE control option.

Note: To use CAIENF/DB2, its SMP/E FMID must be installed.

Note: For information about CAIENF/DB2 requirements, see the *Installation Guide*.

Configure CAIENF/DB2 to Install Intercepts

To configure CAIENF/DB2 to install intercepts in DB2 subsystems, update the CAIENF control option member with the following CAIENF/DB2 control options:

```
MODE(DB2,ON)  
DB2REL
```

DB2REL tells CAIENF to load the CASR230 program module into CSA. The module supports all DB2 levels.

Note: For information about CAIENF/DB2 control options, see the *Reference Guide*.

To ensure that the CAIENF/DB2 intercepts are installed in your DB2 subsystems, issue the START DB2 commands from the CAIENF auto commands member so that the DB2 subsystems start after CAIENF becomes active.

Disable or Enable the Installation of CAIENF/DB2 Intercepts

You can disable or enable the installation of CAIENF/DB2 intercepts in DB2 subsystems that may become active in the future. The operation does not affect currently active DB2 subsystems.

To disable future installation of the intercepts, issue the following command:

```
ENF MODE(DB2,NONE)
```

The next time a DB2 subsystem starts, no intercept is installed.

To enable the installation of the intercepts again, enter the following command:

```
ENF MODE(DB2,ON)
```

CAIENF/DB2 resumes installing intercepts in DB2 subsystems as they start.

Activation of CA Technologies Products (CAIENF/DB2)

CA Technologies products that use CAIENF/DB2 are defined to CAIENF by adding the appropriate DCM statements to the ENFPARMS member. The product routine must be in LNKLST or CAIENF STEPLIB.

Note: For more information, see your product documentation.

CAIENF/USS Operation

CAIENF/USS supports processes executing in the USS environment. It does not have its own address space. It is integrated with the CAIENF base component.

Note: To use CAIENF/USS, its SMP/E FMID must be installed.

Note: For information about CAIENF/USS requirements, see the *Installation Guide*.

When the CAIENF service starts, it activates the CAIENF/USS service.

The following console message appears after CAIENF startup:

```
CARR001I - CA Intercept Technology for z/OS UNIX Services Starting
```

CAIENF/USS requires USS to be initialized before it completes its startup. For example, if CAIENF runs early during system startup before all the USS components have started, CAIENF/USS will wait for those components to start before it continues its own startup. You will see the following message that alerts you that CAIENF/USS is waiting for USS to initialize:

```
CARR008I - Waiting for OMVS Kernel initialization
```

When USS completely initializes, CAIENF/USS continues with its own startup. When startup completes, the following message appears:

```
CARR002I - CA Intercept Technology for z/OS USS initialized, RC=nn
```

If the return code *nn* is not zero, a problem has occurred and CAIENF/USS is not properly initialized. You should correct the source of the problem and reinitialize CAIENF/USS.

Reinitialize CAIENF/USS

You may want to reinitialize CAIENF/USS in order to apply (or remove) maintenance, or to correct an error that occurred during CAIENF/USS startup. You can reinitialize CAIENF/USS using the CARRINIT program.

To use CARRINIT, the user ID must have the following definitions and security permissions for USS:

- Superuser ID (UID 0) or permission to the IBM FACILITY resource BPX.SUPERUSER
- OMVS segment with valid group ID (GID), home directory, and shell program
- Permission to the IBM FACILITY resource BPX.DAEMON if you have defined this resource in your installation

To reinitialize CAIENF/USS using CARRINIT

1. Create a started task procedure using the following JCL statements as an example:

```
//ENFUSS PROC  
//REINIT EXEC PGM=CARRINIT,REGION=0K,TIME=1440  
//STEPLIB DD DSN=step_library
```

The procedure will initialize CAIENF/USS.

2. Submit a batch job that executes the procedure or issue the following command:

```
START ENFUSS  
  
CAIENF/USS initializes.
```

3. Watch for the CARR001I and CARR002I messages to ensure that CAIENF/USS reinitializes properly.

Reinitialize CAIENF/USS by Restarting CAIENF

If you do not have root privileges in USS, you can reinitialize CAIENF/USS by restarting CAIENF. However, restarting CAIENF can disrupt CA Technologies products that require it.

To reinitialize CAIENF/USS by restarting CAIENF, stop CAIENF and then restart it.

SNMP Monitor

The CAIENF SNMP Monitor enables CA Technologies products to issue SNMP traps using a common SNMP proxy. The SNMP Monitor runs as a started task and uses CAIENF to listen for these events. The user ID of the started task must have a valid OMVS segment.

When an ENFSNMP event occurs, it is transmitted using TCP/IP CA Technologies Services, and other SNMP managers can receive the z/OS exception condition. SNMP traps are sent to the port defined for SNMP-trap, which is usually 162. The IP address to which the events are sent is set by the CA Technologies product that issues them and is included in the events.

For information about where to set the IP address, see your product documentation.

Note: The CAIENF SNMP Monitor requires TCP/IP and IBM C libraries.

Start the SNMP Monitor

You start the CAIENF SNMP Monitor to issue SNMP traps using a common SNMP proxy for CA Technologies products.

To start the CAIENF SNMP Monitor

1. Make a copy of the CAWOPROC(ENFSNMPPM) member.
2. Update the copy to suit the requirements of your site.
3. Update the SNMPVARS member. A sample is in CAWOOPTV(SNMPVARS).
4. Update the SNMPCNFG member. A sample is in CAWOOPTV(SNMPCNFG).
5. Move ENFSNMPPM to a system proclib.
ENFSNMPPM is set up as a started task.
6. Add the following command to the CAIENF auto commands member:

```
START ENFSNMPPM
```

The SNMP Monitor will run when CAIENF starts.
7. Stop and restart CAIENF.

Stop the SNMP Monitor

To stop the SNMP Monitor, issue the following command:

```
P ENFSNMPPM
```

Trace and Debug

In general, the TRACE facility traces the general flow and the DEBUG facility provides additional information about the functions it performs. Enter the commands by using the modify MVS console command as follows:

- To turn on TRACE for the CAIENF SNMP monitor, enter the following command:

```
F enfsnmpm, TRAcE ON
```
- To turn off TRACE:

```
F enfsnmpm, TRAcE OFF
```
- To turn on DEBUG for the CAIENF SNMP monitor, enter the following command:

```
F enfsnmpm, DEbug ON
```
- To turn off DEBUG:

```
F enfsnmpm, DEbug OFF
```

Only the capitalized letters are required to invoke the function.

Messages, debug, and trace information are sent to DDNAME CAW1LOG. This must be allocated otherwise file 'DD:CAW1LOG' may be created in the default USS home directory (as specified by the OMVS segment).

Note: For more information about the CAIENF SNMP Monitor Control Options, see the *Reference Guide*.

Environment Variables

The SNMP Monitor runs with a set of environment variables. You can override the default values of these variables by customizing the SNMPVARS parmlib member specified by the ENVVAR DD statement in your ENFSNMPM procedure. A sample is distributed in the CAW0OPTV data set.

SNMP_ECHO_CONFIG_FILE

Specify `SNMP_ECHO_CONFIG_FILE=Y` to echo the configuration file to CAW1LOG. For security reasons the default is not to echo (mainly for SNMPv3 considerations).

Default: `SNMP_ECHO_CONFIG_FILE=N`

CAICATD0000

Specifies the enterprise object identifier (OID).

Default: 1.3.6.1.4.1.791, which is the CA enterprise OID

CAICATD0001

Specifies the system default description, which is the MIB-II machine-specific information for the system. It takes the form of an OID and is used only by the CA Technologies security API.

SNMP_CACHE_TIMEOUT

Specifies the time in seconds to hold valid DNS lookup values in cache (reduce DNS calls). Entries will be re-validated after the time interval.

Default: 600

SNMP_CACHE_WAIT

Specifies the time in to wait after an attempt to resolve a name has failed before attempting again to resolve the name. (Allows for DSN updating.)

Default: 60

SNMP_DEBUG

Indicates whether debugging messages should be issued. You can specify `SNMP_DEBUG=Y` to issue the messages.

Default: `SNMP_DEBUG=N`.

SNMP_TRACE

Indicates whether tracing messages should be issued. You can specify SNMP_TRACE=Y to issue the messages.

Default: SNMP_TRACE=N.

SAPI_ROUTER

Specifies the node name for the CA Audit machine.

Default: none

SAPI_ROUTERPORT

Specifies a fixed port number for CA Audit machine if portmapper not available.

Default: 111

IP_STACK

Specifies the TCP/IP stack to be used. The format is IP_STACK=TCPIP-JOBNAME.

Default: default TCP/IP stack

TZ

Specifies the time zone Affects the timestamp on messages. See the appendix on “Setting the Local Time Zone with the TZ Environmental Variable” in the z/OS UNIX System Services Command Reference.

Configuration File

Beginning with CA Common Services for z/OS Version 14.0, the SNMP Monitor runs with a configuration file. Customize the SNMPCNFG parmlib member specified by the CONFIG DD statement in your ENFSNMPPM procedure. A sample is distributed in the CAW0OPTV data set.

The basic syntax is:

```
node port protocol parameters
```

The full description is contained in the sample. The node corresponds to the “node” in the CAIENF Event which is set by the product creating the Event. Consult the product documentation to determine how the node is set.

If using only the simple SNMP TRAP protocol (as in versions of the SNMP Monitor prior to Version 14.0), a simple one line configuration is all that is necessary. Specify the node as an asterisk, the port with the port number (in versions prior to Version 14.0, this was supplied by the environmental variable SNMP_PORT), the protocol as “snmpv1” and for parameters, specify the SNMP community (in versions prior to Version 14.0, this was supplied by the environmental variable SNMP_COMMUNITY, the default was “public”). See the sample for an example of this entry.

If using SNMPv2 or SNMPv3, see the sample for a full description of the parameters.

For SNMPv3, the ENGINEID can be set for each target host so that the SNMP Monitor can participate in any number of SNMP administratively defined environments. Both MD5 and SHA-1 are supported for authentication; only DES is supported for privacy (encryption).

Chapter 6: Common Communications Interface

This section contains the following topics:

- [Overview](#) (see page 163)
- [Features](#) (see page 164)
- [Configuration](#) (see page 165)
- [Activate CAICCI](#) (see page 170)
- [Customize the CAICCI Service](#) (see page 170)
- [TCP/IP Configurations](#) (see page 170)
- [SNA Configurations](#) (see page 186)
- [z/OS Parallel Sysplex Configurations](#) (see page 190)
- [CAICCI Generic Resources](#) (see page 193)
- [CAICCI SPAWN Facility](#) (see page 194)
- [Assured Delivery](#) (see page 195)
- [Cross Platform Scheduling](#) (see page 200)

Overview

The CA Technologies International Common Communications Interface (CAICCI) provides a messaging infrastructure that enables software products on various platforms to communicate with one another regardless of their communications protocols.

It provides a common approach to CA solution communications requirements. These requirements include the following:

- Product-to-product communication—CAICCI enables CA Technologies products to communicate with other CA Technologies products using a standard protocol.
- Cross-system communication—CAICCI enables CA Technologies products to communicate with other CA Technologies products across systems that support the CAICCI protocols.
- Spawning service—CAICCI enables a CA Technologies product to schedule, execute, and monitor a service program on behalf of a CA solution request.

Features

The CAICCI service provides the following features:

- **Environments Supported Transparently**—CAICCI supports a wide range of communications protocols and network configurations. This enables systems supported by CAICCI to communicate with each other without any special hardware or software.

With CAICCI, software is insulated from the specifics of the environment because it communicates with CAICCI through a platform-neutral API. CAICCI then communicates the information to the targeted platform, handling the specific communications requirements transparently.

The architecture of CAICCI enables it to be extended to support additional platforms and communications protocols without affecting the software solutions.

- **Peer-to-Peer Communication**—Peer-to-peer communication provides services at each end of a communications link that work as equal partners in managing the communication.
- **Transmission Management**—CAICCI completely manages the physical communications between platforms. Applications operate without regard to the details of the underlying communications protocol. CAICCI is multitasking so that applications can use its services concurrently, regardless of any limitations in the underlying network.

In complex networks, the path between applications can include intermediate platforms using different communications protocols. CAICCI automatically selects the optimum path from the source to the target, translating protocols as required.

- **Optional Queuing of Received Data**—Requests may be sent to an application that is busy or is unavailable. CAICCI queues the data until the application is prepared to receive it, either in memory or using a "store and forward" file. It enables messages to be sent to applications even when they are shut down or during network outages, providing a high degree of recoverability and availability.
- **Performance Optimization**—CAICCI optimizes data traffic based on response criteria and system load. It performs load balancing across multiple links and handles data compression. It can be dynamically reconfigured to accommodate changing workloads.
- **Simplified Installation**—Because CAICCI handles messaging for many products, installation involves the definition of only one facility for each computer. This provides a single point of control, a single set of network definitions, and a single facility to monitor and tune. New applications, links, and protocols can be added dynamically.
- **Built-in Diagnostic Aids**—Communications trace facilities are provided to simplify problem determination and resolution. These facilities enable your CA Technologies products to be quickly and consistently supported in the event of any communications problem.

- **Optional Secure Socket Layer (SSL) Support**—SSL support is provided under TCP/IP. Data is encrypted and sent or received across the network providing the maximum level of security.
- **CAICCI Generic Resources**—The feature can group CAICCI applications on different systems into a single logical entity referred to as a CAICCI generic resource. Applications can then communicate using generic names that are mapped to available instances of the specified server application.

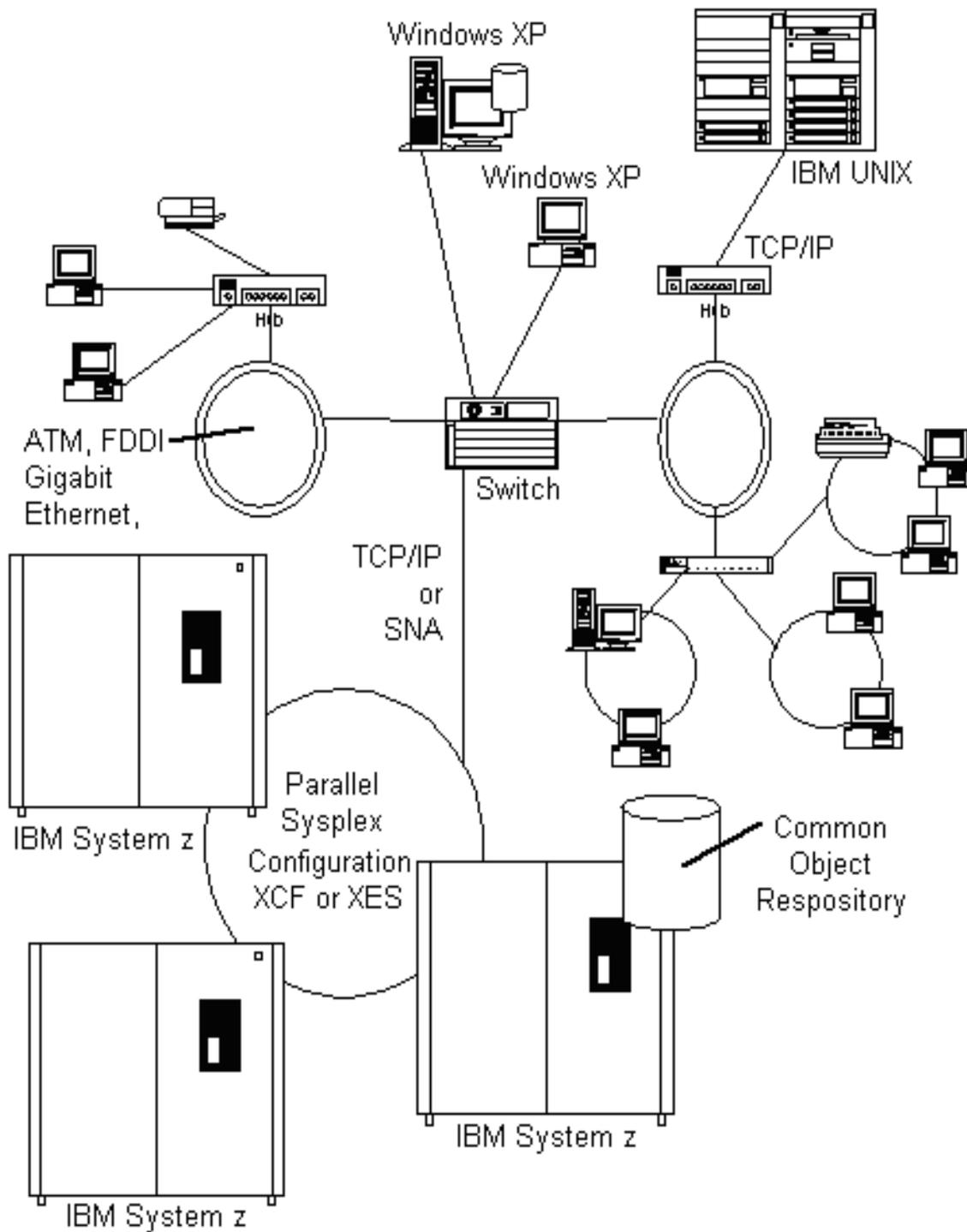
CAICCI generic resources provide scalability, manageability, and fault tolerance. You can add or reconfigure application servers—or to continue processing across network outages—without impacting or reconfiguring client systems.

- **CAICCI SPAWN**—The facility provides transaction management for CAICCI applications. CAICCI applications can use it to trigger and manage units of work as transactions on the mainframe or on other distributed systems. On the mainframe, spawned units of work can exist as subtasks on a server, as CICS transactions, or as stand-alone address spaces.
- **Assured Delivery**—The feature lets you verify whether an intended recipient actually received the message from the sender.

Configuration

CAICCI supports a range of communications protocols and can be configured to adapt to the needs of even the most complex networks.

The following illustration shows a complex network with various platforms running CAICCI over diverse communications protocols.



Communications Protocols

Depending on your workload, network resources, and the combination of CA Technologies products you have installed, you will need to configure one or more of the following protocols:

Cross Memory Services

Provides messaging between applications residing in different address spaces on a single z/OS system. The protocol is enabled whenever CAICCI is running. It is the default protocol used when communications partners reside on the same computer.

Parallel Sysplex Protocols

Provides high-speed communications between z/OS applications running on different computers in a parallel sysplex. For a parallel sysplex configuration, you should deploy the cross-system coupling facility (XCF) or the cross-system extended services (XES) between the z/OS images.

The XCF and XES protocols provide performance, fault tolerance, and scalability to parallel sysplex users.

TCP/IP

Provides CAICCI connections among the following:

- Mainframes running TCP/IP
- PC-to-mainframe client-server applications
- UNIX, Linux, Windows, z/OS, and other CA Technologies platforms

SSL

(CAICCI r12 and above) Provides data encryption and authentication of the connecting systems run over TCP/IP.

SNA (VTAM)

Provides mainframe communications for z/OS, VM, and VSE systems using SNA LUO and interconnects z/OS, VM, and VSE systems using VTAM-supported networking hardware.

How CAICCI Routes Control Information

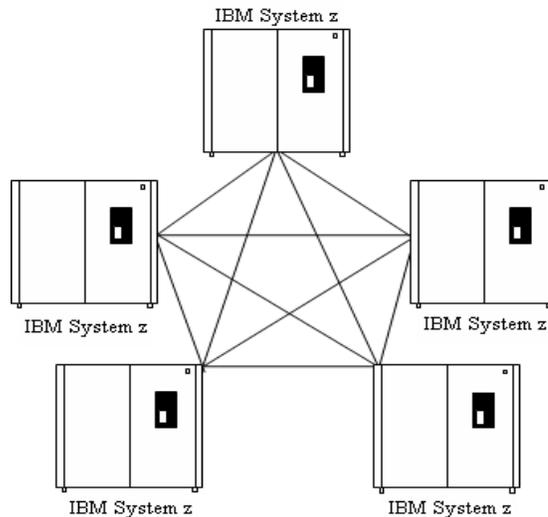
Mainframe CAICCI implementations maintain a replicated, dynamic, real-time directory of applications and network resources to manage communications. Many of the CAICCI internal algorithms require a detailed representation of these objects in order to locate applications and make optimum routing decisions.

Usually, CAICCI maintains the control information for its directory using an ongoing exchange of state change messages. In large or very dynamic networks, transmitting these state changes may consume significant network resources. So, by default, CAICCI only exchanges information among networked computers that are connected directly to one another.

If your networked computers are not connected directly, you can designate one or more CAICCI gateways. Gateways are systems that coordinate state information across CAICCI subnets. A gateway lets CAICCI forward state change messages among computers that cannot communicate with one another directly.

Example: Direct Connections

Five computers are connected directly to one another using CAICCI. They are interconnected using TCP/IP, SNA, or parallel sysplex protocols for communications. In this configuration, CAICCI can maintain accurate state information.



Example: Gateway

Five computers (CPUA, CPUB, CPUC, CPUD, and CPUE) are connected directly to one another using CAICCI. They are interconnected using SNA for communications. The computer, CPUA, has several Windows, UNIX, and Linux computers connected to it. The CA solutions running on Windows, UNIX, and Linux want to communicate with CPUA and CPUB only. You can use the CAICCI GATEWAY control option.

The CAICCI control option member for CPUA can contain the following statements:

```
SYSID(CPUA)
PROTOCOL(VTAM,applid_a,1,CPUA)
PROTOCOL(TCPIP)
GATEWAY(LU0,applid_b,1,CPUB)
NODE(LU0,applid_c,1,CPUC)
NODE(LU0,applid_d,1,CPUD)
NODE(LU0,applid_e,1,CPUE)
CONNECT(CPUB,CPUC,CPUD,CPUE)
```

The CAICCI control option member for CPUB can contain the following statements:

```
SYSID(CPUB)
PROTOCOL(VTAM,applid_b,1,CPUB)
GATEWAY(LU0,applid_a,1,CPUA)
NODE(LU0,applid_c,1,CPUC)
NODE(LU0,applid_d,1,CPUD)
NODE(LU0,applid_e,1,CPUE)
CONNECT(CPUA,CPUC,CPUD,CPUE)
```

The CAICCI control option member for CPUC can contain the following statements:

```
SYSID(CPUC)
PROTOCOL(VTAM,applid_c,1,CPUC)
NODE(LU0,applid_a,1,CPUA)
NODE(LU0,applid_b,1,CPUB)
NODE(LU0,applid_d,1,CPUD)
NODE(LU0,applid_e,1,CPUE)
CONNECT(CPUA,CPUB,CPUD,CPUE)
```

The CAICCI control option members for CPUD and CPUE contain statements similar to those for CPUC.

CAICCI on PCs

On the PC, CAICCI can use *one* of the following configurations:

Client-Server

Is suitable for PC-to-mainframe client-server applications and requires few resources on the PC. The PC communicates with a mainframe CAICCI server using TCP/IP or SSL.

Peer-to-Peer

Provides full CAICCI functionality, which enables the PC to act as a peer. The PC communicates directly with other platforms and can operate without a mainframe CAICCI server, but requires more resources on the PC. Full function CAICCI on a PC always uses TCP/IP as the communications protocol.

Activate CAICCI

You activate CAICCI as a subtask of CAIENF. CAICCI supports communication among local processes through z/OS cross memory services.

To activate CAICCI

1. Specify the following CAICCI control option in the CAICCI control option member:

`SYSID(sysid)`

sysid is a unique eight-character identifier.

2. Concatenate the CAICCI and CAIENF control option members through the ENFPARMS DD statement in the CAIENF started task member.
3. Start or recycle CAIENF.

When CAIENF starts, the SYSID control option activates CAICCI for the applications on the system.

Customize the CAICCI Service

CAICCI control options let you customize the CAICCI service to suit your requirements.

To customize the CAICCI service, specify the CAICCI control options as follows:

- Update the CAICCI control option member.

The changes take effect the next time CAICCI is started (CAICCI runs as a subtask of CAIENF).

- Issue the following command from a console:

```
CCI caicci_control_option
```

The change takes effect immediately.

Note: For information about CAICCI control options, see the *Reference Guide*.

TCP/IP Configurations

CAICCI implements TCP/IP using one or more server address spaces on the mainframe to coordinate processing.

Client-Server

A client-server configuration supports PC-to-mainframe connections. It uses one of the following CAICCI TCP/IP server started tasks. The user ID of the used started task must have a valid OMVS segment. Sample members of the started tasks are distributed in the CAWOPROC data set.

- CCITCP, which is started using the CAICCI PROTOCOL(TCPIP,...) control option
- CCISSL with SSL support, which is started using the CAICCI PROTOCOL(TCPSSL, ...) control option

Although you can start the server task in a client-server configuration using the z/OS START command, it is *not* recommended.

TCP/IP Gateway

A gateway supports the mainframe as a full peer to other platforms in a TCP/IP network. Platforms supported include other mainframe systems, UNIX, Linux, and Windows. It uses one of the following CAICCI TCP/IP server started tasks. The user ID of the used started task must have a valid OMVS segment. Sample members of the started tasks are distributed in the CAWOPROC data set.

- CCITCPGW, which is started using the CAICCI PROTOCOL(TCPIPGW,...) control option
- CCISSLGW with SSL support, which is started using the CAICCI PROTOCOL(TCPSSLGW, ...) control option

You must *not* start the server task for a gateway using the z/OS START command.

Assign TCP/IP Ports

By default, CAICCI uses the following port numbers:

- 1202 for the TCPIP and TCPSSL protocols
- 1721 for the TCPIPGW and TCPSSLGW protocols

If a port is not appropriate, you can use an alternative.

Important! If you use an alternative port, you must ensure that your firewall or TCP/IP configuration permits CAICCI to use that port.

To assign an alternative TCP/IP port, use *one* of the following methods:

- Specify the port number in the CAICCI PROTOCOL statement, for example:

```
PROTOCOL(TCPIP,4500)
```
- Specify the port number through the PARM parameter of the EXEC statement in the CAICCI TCP/IP server started task JCL member. For example, the following statement shows the specification in the CCITCP started task:

```
//CCITCP EXEC PGM=CAS9PDGM,PARM='PORT=4500',...
```

- Specify the port number in your TCP/IP.ETC.SERVICES data set, for example:

```
caicci 4500/tcp
```

You can override this value using one of the two previous methods.

Assign TCP/IP Ports to Multiple Instances of a CAICCI Server

If you must have multiple instances of the CAICCI TCP/IP server in a client-server configuration on a single z/OS system, each requires a unique port number. You can specify the port number in the server started task JCL member using a variable that you override at startup time.

To assign TCP/IP ports to multiple instances of the CAICCI server

1. Use a variable for the port number in the CAICCI TCP/IP server started task JCL member, for example:

```
//CCITCP EXEC PGM=CAS9PDGM,PARM='PORT=&PORT',...
```

The &PORT variable assumes a port number when the server starts.

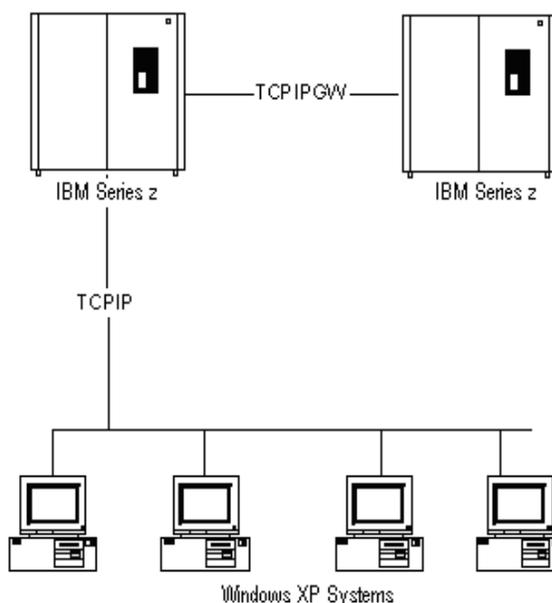
2. Start the server instances, specifying the port numbers, for example:

```
START CCITCP.CCI1,...PORT=4500  
START CCITCP.CCI2,...PORT=4501  
START CCITCP.CCI3,...PORT=4502
```

The server instances start, using the specified port numbers.

Example: Configuration with Two z/OS Systems and Multiple PCs

The following illustration shows a TCP/IP configuration with two connected z/OS systems and multiple PCs connected to one system using CAICCI TCP/IP client-server:



The configuration uses the following CAICCI TCP/IP protocols:

- TCPIP to let the z/OS systems communicate with one another
- TCPIP to support PC client-server applications

The CAICCI control option member on the respective z/OS system contains the following CAICCI control options:

z/OS System	CAICCI Control Option
A	SYSID(CCIA) PROTOCOL(TCPIP) PROTOCOL(TCPIP)GW NODE(TCPIP)GW,CCIB.COM,01,CCIB,8192,START/SHUT CONNECT(CCIB)
B	SYSID(CCIB) PROTOCOL(TCPIP) PROTOCOL(TCPIP)GW NODE(TCPIP)GW,CCIA.COM,01,CCIA,8192,START/SHUT CONNECT(CCIA)

SYSID

Defines CAICCI on the z/OS system.

PROTOCOL

Specifies the CAICCI TCP/IP protocols.

For SSL, the PROTOCOL control options are similar to the following:

```
PROTOCOL (TCPSSL, 1202, 1, CCIx, 16384)
PROTOCOL (TCPSSLGW, 1721, 1, CCIx, 16384)
```

NODE

Specifies the remote node.

To use the logical host names, CCIA.COM and CCIB.COM, the TCP/IP environment must be able to reference a TCP/IP domain name server (DNS) or has a host name table that can translate the specified names to IP addresses.

For best performance, the data packet size (8192 in this example) should be slightly smaller than the maximum transmission unit (MTU) supported by your TCP/IP configuration.

CONNECT

Connects the system to the remote node.

Note: For more information about CAICCI control options, see the *Reference Guide*.

Note: Similar configuration is suitable for connecting other full-function CAICCI platforms to the mainframe, including UNIX and Windows.

For the PC-to-mainframe client-server configuration, no further changes are required on the mainframe. In a client-server configuration, the remote PCs initiate the connections, which require CAICCI be configured on the PCs.

Customize the CAICCI TCP/IP PROC Names

To change the default names of the CAICCI procs to conform to site naming standards, you can use the CAICCI SPNPARMS DD. You can add parms similar to the following:

```
TCPIP      SERVICE  SERVER_NAME=MVS_START_SERVER,
           MAX#_PROCESSES=1
           PROCESS  PROCESS_TYPE=MVS_STC, PROCNAME=xxxxxxxx,
           TERM_TYPE=STOP, PAUSE_TIME=28,
           PARM=''''
```

The label (starting in column 1) identifies the protocol associated with the proc. The label is the protocol (first) parameter of the PROTOCOL statement. You only need to update the PROCNAME field with the name you wish to use. Do not change any other PARM, except for possibly the PAUSE_TIME parameter.

The new PROCESS parameter TERM_TYPE directs how the STC Spawn Server is to terminate the Started Task. Specifying TERM_TYPE=CANCEL, which is the default, directs the STC Spawn Server to terminate the Started Task using the CANCEL command. Specifying TERM_TYPE=STOP directs the STC Spawn Server to terminate the Started Task using the STOP command.

The new PROCESS parameter PAUSE_TIME=xx directs the STC Spawn Server to wait xx seconds for the Started Task to respond to a STOP command. If the Started Task has not terminated within xx seconds, a CANCEL command is then issued. The PAUSE_TIME value ranges from 4 to 60 seconds, in multiples of four. The default value is 10 seconds.

The PAUSE_TIME parameter may be increased for any newly added or existing SPNPARM definitions if the Started Task consistently fails to terminate before the STC Spawn Server reverts to issuing a CANCEL command.

CCISPNPM is supplied in the CAWOOPTN data set as a sample SPNPARMS member.

Configure CAICCI on Windows Running CA Technologies Products

Full CAICCI functionality is available on a Windows system running CA Technologies to support a peer-to-peer configuration.

To configure CAICCI on a Windows system running CA Technologies products

1. Ensure that you can ping the mainframe system to which you want to connect.

If you cannot ping the mainframe system, review the TCP/IP setup on the Windows PC and the mainframe. You may need to add an entry for the z/OS host name in DNS or the `\system32\drivers\etc\hosts` file on the PC.

2. Define a connection between the Windows PC and the mainframe system using [LOCAL and REMOTE statements](#) (see page 177) in the `install_path\CAIUSER\ccirmtd.rc` CAICCI configuration file.

3. Issue the following command to determine whether the CAICCI Remote Server is installed:

```
ccicntrl
```

A series of messages appears including the following if the Remote Server is installed and running:

```
Service "CA-Unicenter (Remote)", STATUS is "Running"
```

4. If the Remote Server is running and you updated the CAICCI configuration file in Step 2, issue the following commands:

```
ccicntrl stop rmt  
ccicntrl start rmt
```

The Remote Server stops and then restarts. You have finished configuring CAICCI on the Windows system.

5. If the Remote Server is not installed, issue the following command:

```
ccicntrl install rmt install_path\bin
```

The Remote Server is installed.

6. If the Remote Server is not running, issue the following command:

```
ccicntrl start rmt
```

The Remote Server starts. You have finished configuring CAICCI on the Windows system.

LOCAL and REMOTE Statements

The LOCAL statement applies to the local computer. The REMOTE statements apply to the remote nodes that exchange information with CAICCI.

The statements have the following syntax:

```
LOCAL|REMOTE=tcp/ip_name cci_name buffersize STARTUP|NOSTART [ALIAS=alias_name]  
[PORT=port_number] retry_interval
```

tcp/ip_name

Specifies either an IP address in IPV4 format (dotted decimal notation: ddd.ddd.ddd.ddd), in IPV6 format ("xx:xx:xx:xx:xx:xx:xx:xx") in release r12 and above, or a name as input to a name service to retrieve an IP address. You can use *tcp/ip_name* with the ping command to determine whether a remote connection is live.

Default: Local TCP/IP host name

cci_name

Specifies the host name, which may or may not be the same as the TCP/IP host name. CAICCI uses *cci_name* to identify this host.

For the REMOTE statement that defines the mainframe, the CAICCI name is the value specified by CAICCI SYSID(*sysid*) control option on the mainframe.

Default: Local host name

buffersize

Specifies the maximum size of the buffer CAICCI receives or sends. It is used for segmenting data transfer.

Default: 32768

Limits: 1024 through 32768

Important! Contact CA Support before changing the buffer size.

STARTUP|NOSTART

Specifies whether to initiate a remote connection when CAICCI becomes active:

- STARTUP specifies that CAICCI will initiate the connection.
- NOSTART specifies that the remote system will initiate the connection.

ALIAS=*alias_name*

(Optional) Defines an alias name to differentiate multiple remote systems that have the same first eight characters in their host names.

PORT=*port_number*

(Optional) Specifies the port number.

Default: 1721

retry_interval

Specifies the number of seconds between attempts to connect:

- -1 specifies that attempts to connect starts with a two-second retry interval that doubles after each unsuccessful attempt.
- 0 specifies that no attempts are made to connect.
- Integer greater than zero specifies that attempts to connect occur at the specified retry interval.

Example: LOCAL Statement

The following LOCAL statement tells CAICCI that the TCP/IP name for the local system is NTSRVR1 and that any remote system wanting to communicate with this system can do so by referencing NTSRVR1 as the name.

```
LOCAL=NTSRVR1 NTSRVR1 32768 STARTUP
```

Example: REMOTE Statement

The following REMOTE statement tells CAICCI to attempt to connect to 172.22.111.121 and to register MF01 as the CAICCI name internally.

```
REMOTE=172.22.111.121 MF01 32768 STARTUP PORT=1721
```

Example: REMOTE Statement Using an Alias

The following REMOTE statement tells CAICCI to attempt to connect to the system whose TCP/IP name is NTSRVR1 and whose CAICCI name (host name) is WINNTSERVER1. It also lets CAICCI to send and receive messages using the alias NT1.

```
REMOTE=NTSRVR1 WINNTSERVER1 32768 STARTUP ALIAS=NT1
```

Issue CAICCI TCP/IP Gateway Commands

CAICCI TCP/IP gateway commands let you control the connections in and retrieve information about a gateway configuration.

To issue a CAICCI TCP/IP gateway command, use the z/OS MODIFY command for the appropriate gateway server started task:

```
F CCITCPGW,caicci_gateway_command
```

```
F CCISSLGW,caicci_gateway_command
```

ACTIVE Command—Vary the Connection State of a Remote CAICCI to the Online State

The ACTIVE command changes the connection state for a remote host from OFFLINE to INACTIVE/CONNECTABLE. It can optionally initiate the connection to the remote system and provide a different port number. A port number is required if no NODE/GATEWAY statement had been defined for the remote system.

This command has the following format:

```
A[ctive],Sysid[,Connect[,portnumber]]
```

sysid

Identifies the remote CAICCI that you want to activate.

Connect

Forces a connection to be made to the remote CAICCI.

portnumber

Identifies the port number of the remote CAICCI that is to be used when connecting.

AUTOPRINTT Command—Enable/Disable AUTOPRINTT

The AUTOPRINTT command turns on and off the automatic PRINTT of traces after an exceptional event. Having the internal trace table printed immediately after an event occurs provides the most recent tracing entries leading up to the event.

This command has the following format:

```
AU[toprintt][,Yes | No | On | Off]
```

BUILD Command—Display Build Date and Time

The BUILD command displays the build date and time for this module.

This command has the following format:

```
B[uild]
```

CONNECT Command—Initiate a Connection to a Remote CAICCI

The CONNECT command resets the DISCONNECT command's override of the retry time for the remote system. It resets the retry time to its original value. The gateway server started task tries to connect to the disconnected remote system based on this original retry time. Optionally specify a port number to initiate the connection to the remote system using the different port number. A port number is required if no NODE/GATEWAY statement is defined for the remote system.

This command has the following format:

```
C[onnect] ,sysid[ ,portnumber]
```

sysid

Identifies the remote CAICCI to connect to.

portnumber

Identifies the remote CAICCI port number to use when connecting.

DISCONNECT Command—Disconnect from a Remote System

The DISCONNECT command disconnects the host from a remote system. It sets the connection retry time for the remote system to zero (0). The gateway server started task will no longer try to reestablish the connection, however a connection from the remote system will still be accepted.

This command has the following format:

```
DISCON[nect] ,sysid
```

sysid

Identifies the remote CAICCI from which you want to disconnect.

DUMP Command—Terminate and Dump a Connection

The DUMP command terminates and dumps the underlying task of the thread that owns the *sysid*.

This command has the following format:

```
DU[mp] ,sysid
```

sysid

Identifies the remote CAICCI with the connection task to terminate and dump.

HELP Command—Display Help for Commands

The HELP command displays help for one or all available commands.

This command has the following format:

H[elp] [, *command*]

command

Identifies the single command for which help is requested. If no command is specified, help displays for all available commands.

INACTIVE Command—Vary the Connection state of a Remote CAICCI to the Offline State

The INACTIVE command shuts down a connection or cancels an ongoing retry to a remote CAICCI. The sysid is now in an OFFLINE state. The gateway server started task no longer initiates a connection to the remote CAICCI and a connection from the remote CAICCI is not accepted.

This command has the following format:

I[nactive] , *sysid*

sysid

Identifies the sysid to put in an OFFLINE state.

KILL Command—Terminate a Connection

The KILL command terminates the underlying task of the thread that owns the Sysid. No dump is generated.

This command has the following format:

K[ill] , *sysid*

sysid

Identifies the remote CAICCI whose connection task is to be terminated.

NETSTAT Command—Display TCP/IP Connection Status

The NETSTAT command displays the TCP/IP connection status of one or all sysids.

This command has the following format:

N[etstat] [, ALL | *sysid*]

sysid

The sysid to display the TCP/IP connection status for.

NOTRACE Command—Turn Off Tracing

The NOTRACE command turns off active tracing, which includes the internal trace.

This command has the following format:

```
NO[trace]
```

PING Command—Check Communications

The PING command checks whether communications are established with a CAICCI TCP/IP server. You should receive three messages similar to the following for each PING command:

```
CAS9899I CCI Ping requested to socket A97S  
CAS9899I CCI Ping sent to socket A97S  
CAS9899I CCI Ping response received from socket A97S
```

The command has the following syntax:

```
P[ing],sysid
```

sysid

Identifies the remote CAICCI to be pinged.

RECYCLE Command—Recycle a Connection

The RECYCLE command terminates the connection and initiates or allows an immediate reconnection to the specified remote CAICCI.

This command has the following format:

```
REC[ycle],sysid
```

sysid

Identifies the remote CAICCI whose connection is to be recycled.

RELEASE Command—Display CAICCI Release Level

The RELEASE command displays the release, genlevel and system info for local and remote CAICCI hosts

This command has the following format:

```
R[elease], [ALL | sysid]
```

Note: Providing no parameter displays the local host only.

ALL

Specifies that you want to display the release levels of CAICCI for the local host and for all connected remote hosts.

sysid

Identifies the CAICCI for which you want to display the release level.

REPORT Command—Display CAICCI Name and Routing Table

This command displays the CAICCI name and routing table.

This command has the following format:

```
REP[ort]
```

STATUS Command—Display CAICCI Connection Status

The STATUS command displays the CAICCI connection status of one or all sysids.

This command has the following format:

```
S[tatus][, ALL | sysid]
```

ALL

Displays the connection status for all sysids.

sysid

Displays the connection status for a specific sysid.

SVCDUMP Command—Take an SVC Dump of the Address Space

The SVCDUMP command takes an SVC dump of the address space of the gateway server started task address space and continues.

This command has the following format:

```
SV[cdump]
```

SYSDTECT Command—Turn Sysid Detection On and Off

The SYSDTECT command turns on and off the display of new CAICCI sysids that the local CAICCI detects through its direct connections or gateway routing.

This command has the following format:

```
[SY]sdtect[,Yes | No | On | Off]
```

Yes/On

Turns on the display of new CAICCI sysids.

No/Off

Turns off the display of new CAICCI sysids.

TRACEON Command—Turn On Tracing

The TRACEON command turns on active tracing and writes it to one of the following: the system log, the console, the internal trace table, or to a specified ddname.

This command has the following format:

```
Traceon(,WTL | WTO | INT | ddname)
```

WTL

Writes the trace to the system log.

WTO

Writes the trace to the console.

INT

Writes the trace to the internal trace table.

ddname

Writes the trace to the specified *ddname*. If the *ddname* is not in the jobstream, it is dynamically allocated.

TRCROUTE Command—Turn Route Tracing Detection On and Off

The TRCROUTE command turns on and off the display of new CAICCI routes that the local CAICCI detects through its direct connections or gateway routing.

This command has the following format:

```
[TRC]route[,Yes | No | On | Off]
```

Yes/On

Turns on the display of new CAICCI routes.

No/Off

Turns off the display of new CAICCI routes.

WAKE Command—Wake a Connection Task

The WAKE command posts the underlying task of the task that owns the Sysid.

This command has the following format:

```
W[ake] ,sysid
```

sysid

Identifies the remote CAICCI whose connection task is to be posted.

XNTREPORT Command—Display CAICCI Name and Routing Table

The XNTREPORT command displays the CAICCI name and routing table.

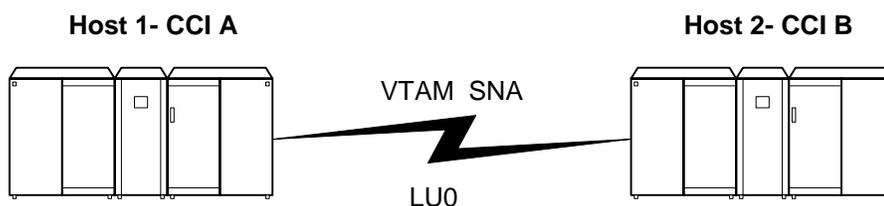
This command has the following format:

```
X[ntreport]
```

SNA Configurations

In CAICCI SNA network configurations, two or more mainframe CAICCI instances are connected using VTAM and SNA LU0 protocols. The host computers may be local or remote and may be running z/OS, VM, or VSE operating systems. Each host may run any number of CA Technologies products.

The following graphic shows a typical two-host CAICCI network configuration where each host runs one CA Technologies product. The same concepts apply, no matter how many CA Technologies products share CAICCI connections.



The two mainframes are referred to as CCIA and CCIB. In each computer, you must configure VTAM resources for the local and remote CAICCI instances using the following process:

1. Define VTAM resources
2. Configure CAICCI

Define VTAM Resources

The procedure uses the illustration in [SNA Configurations](#) (see page 186) as an example. It uses CCIA and CCIB as host and VTAM resource names; however, you may use names appropriate to your installation and any naming standards you have adopted. In many cases, you may use z/OS system symbols in your VTAM definitions, enabling a single set of definitions to be shared across multiple systems.

To define VTAM resources

1. Define the local CAICCI VTAM applications on each computer.

Define CCIA as follows:

```
CCIH0ST1 VBUILD TYPE=APPL
CCIA     APPL  ACBNAME=CCIA,
          AUTH=(ACQ,NOTSO)
```

Define CCIB as follows:

```
CCIH0ST2 VBUILD TYPE=APPL
CCIB     APPL  ACBNAME=CCIB,
          AUTH=(ACQ,NOTSO)
```

Depending on your network configuration and site standards, you may also want to include other VTAM parameters such as CERTIFY, EAS, and VPACING.

Note: For information about VTAM parameters, see your VTAM documentation.

2. If you are not using advanced peer-to-peer networking (APPN) with DYNLU=YES, define a VTAM cross-domain resource for each remote CAICCI system that is to be accessed from CCIA and CCIB.

Note: If you are using SNA exclusively for PC-to-mainframe connectivity, you can bypass this step.

Define the following VTAM cross-domain resource on host CCIA:

```
CCICDRS1 VBUILD TYPE=CDRSC
CCIB     CDRSC  CDRM=CDRMB, ISTATUS=ACTIVE
```

Define the following VTAM cross-domain resource on host CCIB:

```
CCICDRS2 VBUILD TYPE=CDRSC
CCIA     CDRSC  CDRM=CDRMA, ISTATUS=ACTIVE
```

3. Activate the new VTAM resources.

On CCIA, issue the following commands:

```
V NET,ACT,ID=CCIH0ST1
V NET,ACT,ID=CCICDRS1
```

On CCIB, issue the following commands:

```
V NET,ACT,ID=CCIH0ST2
V NET,ACT,ID=CCICDRS2
```

4. Add these new resource definitions to your VTAMLST ATCCONxx member so that they become active when VTAM is initialized.
5. Define other VTAM resources such as cross-domain resource manager (CDRM) major nodes, channel-to-channel (CTC) devices, and network control program (NCP) major nodes for your configuration.

Note: For information about VTAM resource definitions, see your VTAM documentation.

Note: Before CAICCI can use VTAM generic resources, you must have performed the setup outlined in your VTAM documentation.

Configure CAICCI for SNA

After you define the VTAM resources, you configure CAICCI to use those resources.

To configure CAICCI for SNA

1. Update the CAICCI control option member:

Note: For more information about CAICCI control options, see the *Reference Guide*.

- SYSID(*sysid*) to identify CAICCI on each host uniquely in the network.
- PROTOCOL(VTAM,*applid*,...) to configure CAICCI on each host to use VTAM as the network protocol. Use the ACB name specified for the VTAM resource defined for CAICCI for *applid*.
- (Not required if you are using SNA exclusively for PC-to-mainframe connections) NODE(LU0,*applid*,...) to define each remote host that is to be connected at startup.
- CONNECT(*sysid*) to connect CAICCI to the remote host.
- GENERIC(VTAM,*generic_name*) if you are using CAICCI as a VTAM generic resource. *generic_name* is the name that remote users specify to connect to this CAICCI host.

Using the illustration in [SNA Configurations](#) (see page 186) as an example, the CAICCI control options for the hosts are as follows:

z/OS System	CAICCI Control Option
Host 1	SYSID(UNIQCCIA) PROTOCOL(VTAM,CCIA,01,UNIQCCIA,4096,START/SHUT) NODE(LU0,CCIB,01,UNIQCCIB,4096,START/SHUT) CONNECT(UNIQCCIB)

z/OS System	CAICCI Control Option
Host 2	SYSID(UNIQCCIB) PROTOCOL(VTAM,CCIB,01,UNIQCCIB,4096,START/SHUT) NODE(LU0,CCIA,01,UNIQCCIA,4096,START/SHUT) CONNECT(UNIQCCIA)

You have defined the CAICCI SNA network.

2. Start CAIENF on both hosts.

The CAICCI SNA network becomes active.

z/OS Parallel Sysplex Configurations

If you operate z/OS in a parallel sysplex configuration, CAICCI can take advantage of XCF and XES hardware to provide high performance messaging between members of the sysplex. By using XCF or XES, you can achieve very high bandwidth connections between z/OS systems in a sysplex, without the need for a network protocol such as TCP/IP or VTAM.

XCF and XES also support numerous fault tolerance features, enabling communications to continue even following what otherwise might be a disastrous hardware error. Together, these features result in faster, more reliable communications; they are especially recommended for those environments with large application workloads.

XCF provides efficient, high-bandwidth communications using a technique known as XCF signaling. With XCF, you can configure coupling facility paths or CTC connections for cross-system communications. CAICCI message traffic is directed over an XCF signaling path according to how you configure XCF and CAICCI. XCF requires appropriate hardware and software definitions.

XES enables CAICCI to manage structures in a hardware device known as a coupling facility for messaging. Like XCF, XES provides high bandwidth. It is implemented by a series of coupling facility operations. XES requires an appropriate coupling facility hardware and software configuration.

If the sysplex does not include a coupling facility or when adequate coupling facility resources for CAICCI do not exist, you can use XCF. If adequate coupling facility resources are available for CAICCI, you can use the XES protocol to exploit the coupling facility structures, yielding important performance benefits.

Use PROTSEC to specify XCF and/or XES type links as secure connections. The parameters passed to PROTSEC specify secure links over XCF or XES protocols. You can specify either parameter or both parameters in either order as shown in the following examples:

```
PROTSEC(XCF,XES)
```

```
PROTSEC(XES,XCF)
```

```
PROTSEC(XCF)
```

```
PROTSEC(XES)
```

Note: PROTSEC does not actually enable the flow of encrypted data; PROTSEC internally flags the XCF/XES link as a secure link.

Note: For more information about setting up XCF or XES, see IBM's *z/OS MVS Setting Up a Sysplex* and *z/OS MVS Programming Sysplex Services Guide*. For more information about the PROTSEC command, see the *Reference Guide*.

Configure CAICCI for XCF

To configure CAICCI for XCF

1. Ensure that the XCF hardware and software are installed and operational. If you are not using XCF for other purposes, you need to set up sysplex couple data sets and signaling paths. Any hardware acceptable to XCF can be used with CAICCI, including CTC and coupling facility signaling paths.
2. Examine your SYS1.PARMLIB(COUPLExx) member to ensure that you have proper sysplex definitions; each z/OS system must be part of the same sysplex and have couple data sets, transport classes, and signaling paths defined.

Note: For information about how to set up XCF signaling, see IBM's *z/OS MVS Setting Up a Sysplex*.

3. Ensure that CAICCI on each host has a unique identifier assigned by the following control option:

`SYSID(sysid)`

4. Specify the following statement in the CAICCI control option member:

`PROTOCOL(XCF,group_name)`

group_name

Specifies the name of the logical XCF group you want CAICCI to use.

CAICCI is configured to use XCF as a protocol.

With these definitions in place, CAICCI will use XCF services to communicate among z/OS systems in the sysplex. If you anticipate very high volumes of activity, you may want to monitor XCF activity periodically. CA SYSVIEW Performance Management and similar performance monitors can provide this information.

Configure CAICCI for XES

To configure CAICCI for XES

1. Ensure that you have the prerequisite hardware and software enabled. z/OS should be configured to operate in a parallel sysplex, with one or more coupling facilities.

Note: For information about how to set up XES and coupling facility structures, see IBM's *z/OS MVS Setting Up a Sysplex*.

Use the IBM IXCMIAPU administrative data utility to define the CAICCI coupling facility structure.

Note: For information about the IXCMIAPU administrative data utility, see IBM's *z/OS MVS Setting Up a Sysplex*.

Your input should be similar to this:

```
STRUCTURE NAME(name)  
SIZE(size)  
PREFLIST(cf_name)
```

name

Defines the name of the structure you want CAICCI to use.

size

Specifies the size of the structure in 4-KB units. Typically, a value of 1000 (4 MB) should suffice, but the optimum size depends on your workload and communication volume.

cf_name

Specifies the name of the coupling facility in which you want the structure defined.

2. Ensure that CAICCI on each host has a unique identifier assigned by the following control option:

```
SYSID(sysid)
```

With these definitions in place, CAICCI will use XES services to communicate among z/OS systems in the sysplex. You may want to monitor the CAICCI coupling facility structure periodically to ensure that it is of adequate size. CA SYSVIEW Performance Management and similar performance monitors can provide this information.

Note: If you define an LU0 node and connect for a system that may also connect using XCF or XES, the first connection to complete will be the active one. The other protocol will cancel its attempt. This may cause the following messages to appear in your CAIENF job log:

```
CAS9603I - CAICCI Group CCIXCF from System @@@@  
CAS9602E - CAICCI SYSID sysid could not be logged on
```

These messages will recur as part of normal recovery and not cause problems with CAICCI or the products using it.

CAICCI Generic Resources

In a sysplex environment, you can configure multiple CAICCI instances to act as a CAICCI generic resource. CAICCI generic resources let you declare a group of CAICCI instances that are to be treated as a single logical entity, providing scalability, fault tolerance, and manageability benefits.

Consider a situation where CA IDMS/DB is running on five z/OS systems in a sysplex configuration. The individual systems are named CCI1 through CCI5, and each of these systems runs a database server.

Without CAICCI generic resources, CA IDMS/DB clients connecting to the mainframe specify the host name of a server to connect to. In this configuration, assigning particular users to specific database server instances is handled on a case-by-case basis; ongoing management can be a burden because adapting to changing workloads requires you to manually reconfigure client systems. Reconfiguring to achieve load balancing or to accommodate other mainframe changes is difficult, especially if large numbers of client systems are involved.

With a CAICCI generic resource, you assign a logical name to a collection of mainframe resources, avoiding many of the problems outlined previously. Clients connect using this logical name rather than a specific host name. You can reconfigure your mainframe resources without having to reconfigure client systems. You can adapt to changing mainframe requirements by simply changing server configurations.

Note: CAICCI generic resources works independently from CAICCI support for VTAM generic resources, and you can use either or both.

Configure CAICCI Generic Resources

Important! Certain CA Technologies products have unique system affinity requirements and must guarantee that all communications occur with a specific instance of an application. Before using CAICCI generic resources, see the product documentation to ensure that a product supports CAICCI generic resources.

To configure a CAICCI generic resource, specify the following CAICCI control option:

`SYSPLEX(name)`

name

Specifies the logical name of the CAICCI generic resource associated with this CAICCI instance.

Limits: 1 through 8 characters

When the definition is in place, applications can connect using the logical name rather than a specific host name.

CAICCI SPAWN Facility

The CAICCI SPAWN facility enables certain CA Technologies products to trigger and manage units of work as transactions under CAICCI control. Using CAICCI and the CAICCI SPAWN facility, applications can do the following:

- Trigger diverse types of work on many platforms, ranging from CICS transactions and started tasks to processes on a variety of z/OS, VSE, or UNIX platforms.
- Monitor transactions initiated by the CAICCI SPAWN facility.
- Exchange data with spawned transactions.

Configure CAICCI SPAWN Facility

You need to configure the CAICCI SPAWN facility if your CA Technologies products use it.

If you are spawning CICS transactions, you must install and configure CAIENF/CICS SPAWN.

To configure the CAICCI SPAWN facility

1. Create a CAICCI SPAWN parameter member in the CAWOOPTN data set with the parameters required by your CA Technologies products.
Note: For information about product-specific CAICCI SPAWN parameters, see your product documentation.
2. Add a SPNPARMS DD statement in the CAIENF started task JCL member, and point it to the CAICCI SPAWN parameter member.

Assured Delivery

Some CA Technologies products require Assured Delivery. It is a feature that monitors conversations between CA solutions located on local and remote hosts in a CAICCI network. It stores the sent and received data in a database that can be used to verify whether an intended recipient actually received the data from the CA solution that sent it.

Define the Database

Assured Delivery requires a VSAM database (also known as the LOGGER database) to store information and data being sent and received between CA Technologies solutions defined to the CAICCI network.

The Assured Delivery service is started using the LOGGER control option and can be placed in the CCIPARMS or entered as a z/OS console command. For more information on the LOGGER control option, see the *Reference Guide*.

To define the database for Assured Delivery

1. Customize the CAI.CAW0JCL(LOGRALOC) member to suit your requirements:
 - Ensure that the NAME parameters specify the high-level qualifier that is used for your installation.
 - Customize the CYLINDERS parameter to suit the requirements for your CA Technologies products.
 - Update the VOL parameter. You can specify multiple volumes.
 - (Optional) Customize the CISZ parameter for the data and index components.

```
DEFINE CLUSTER -  
  NAME(xxxxxxxx.CCLOGER) -  
  SHR(3,3)  
  RECORDSIZE(4096 33000) -  
  CYLINDERS(n,m) -  
  SPANNED -  
  
  REUSE -  
  KEYS(38 0)  
  VOL(vvvvvv,...) -  
DATA -  
  (NAME(xxxxxxxx.CCLOGER.DATA) -  
  CISZ(4096)) -  
INDEX -  
  (NAME(xxxxxxxx.CCLOGER.INDEX) -  
  CISZ(3584))
```

xxxxxxxx

Replace with a VSAM data set high-level qualifier suitable for your installation.

n, m

The size of the VSAM file (*n* and *m*) varies by application and anticipated logging volume. Consult your CA Technologies product documentation for details on how to specify these parameters.

Important! You cannot share the VSAM cluster across systems.

yyyyy

Specify multiple volumes and the DATA and INDEX control interval size (CSIZ) may be modified, depending on the average data length.

2. Submit the LOGRALOC job.

IDCAMS creates the database.

Activate Assured Delivery

The LOGGER control option is used to activate the Assured Delivery feature of CAICCI and specify the number of VSAM strings and buffers that will be saved in the LOGGER database.

Parameters that must be entered for LOGGER are as follows:

strings

The number of VSAM strings. Calculate this number as follows: $2 \times \text{\#systems} + 5$

The default is 20 strings.

buffer1

The number of VSAM data buffers. Calculate this number as follows: $2 \times \text{\#systems} + 3$

The default is 12 data buffers.

buffer2

The number of VSAM index buffers. Calculate this number as follows: \#systems but not < 12

The default is 12 index buffers.

reorg

Y or N. If Y is specified, LOGGER will reorganize the VSAM database at start-up of CCILGR.

The default is N.

Example (console)

```
CCI LOGGER(25,23,12,Y)
```

Example (ENFPARMS)

```
LOGGER(25,23,12,Y)
```

To ensure that Assured Delivery is activated automatically as part of CAIENF startup, add the CAICCI LOGGER control option in your CAICCI control option member.

View the Database Contents

To view the contents of the database, use the z/OS MODIFY command:

```
F CCILGR,logger_database_command
```

DBSTATUS Command—Display Database Activity

The DBSTATUS command displays the following information about the database:

- Number of successful look-asides
- Number of buffer reads
- Maximum number of strings
- Available space in KB
- Number of extents currently allocated

DISPLAY Command—Display Space Information

The DISPLAY command displays the following information about the database:

- Available space in KB
- Percent full
- Number of extents currently allocated

PURGE Command—Purge Data from the Database

The PURGE command purges specified data from the database.

The command has the following syntax. All parameters are optional but position dependent. If more than one parameter is present, the command evaluates them using AND operations to determine which records are to be purged.

PURGE, *sysid*, *applicationid*, *startdate*, *starttime*, *enddate*, *endtime*

sysid

Specifies a CAICCI ID. Records associated with this CAICCI will be purged.

applicationid

Specifies an application ID. Records associated with this application will be purged.

startdate and ***starttime***

Specifies the beginning of a time interval over which records will be purged.

startdate is in *yyyddd*, where *yyyy* is the year and *ddd* is the day of the year (for example, 2006305).

starttime is in *hh:mm:ss*.

Default: 0

enddate and ***endtime***

Specifies the end of a time interval over which records will be purged.

enddate is in *yyyddd*, where *yyyy* is the year and *ddd* is the day of the year (for example, 2006305).

endtime is in *hh:mm:ss*.

Default: 999999

Example: Purge Specific Records

The following command purges the records from November 1, 2004 at 8 a.m. through November 30, 2004 at midnight that is associated with the CAICCI identified as A01IENF and the CA-TSS application:

```
F CCILGR,PURGE,A01IENF,CA-TSS,2004305,08:00:00,2004334,24:00:00
```

Example: Purge All Records

The following command purges all records from the database:

```
F CCILGR,PURGE,, ,0
```

RELEASE Command—Display Database Creation Time

The RELEASE command displays the date and time when the database is created.

REPORT Command—Generate Report

The REPORT command allocates a data set using the following DD statement:

```
ADREPORT DD SYSOUT=A,HOLD=YES
```

The command then writes a summary of the records in the database to this data set.

STATUS Command—Display Conversation Status

The STATUS command lists the sub-tasks, system IDs, and applications to which data is to be sent. It also lists the number of conversations attempted and completed since the last time Assured Delivery was activated.

Cross Platform Scheduling

CA Technologies scheduling engines can request that their work be run by another CA Technologies scheduling engine on a different platform. This is known as cross platform scheduling. For example, CA 7 Workload Automation can request that a CA NSM Workload job be run.

This section describes some considerations for the XPS client, and CAICCI and z/OS connections when using cross platform scheduling.

XPS Client

The workload can be thought of as primarily under the control of the XPS client. The XPS client requests work and monitors status for the purposes of workload control. The XPS server only acts to initiate work and to communicate status information about the work to the XPS client.

Each scheduling engine documents how to start the XPS piece. You will need to check those product guides to ensure you have all the configuration and processes started to make XPS work. This guide only describes how to set up CA Common Services.

CAICCI Connections

To participate in cross platform scheduling, all machines must have CAICCI installed, active, and connected to the machines involved.

For non z/OS communications, CAICCI connections are defined in the following files:

- UNIX- \$CAIGLBL0000/ci/config/<machine name>/ccirmtd.prf
- Windows-\TND\CAIUSR\ccirmtd.rc

The format of the REMOTE control cards are:

```
REMOTE = ip address cciname blocksize [STARTUP] [PORT=nnnn] [RETRY=x]
```

ip address

Specifies the numerical TCP/IP address of the node or the hostname that can be resolved into a TCP/IP address. You can specify an IP address in either IPV4 format (dotted decimal notation: "ddd.ddd.ddd.ddd") or IPV6 format ("xx:xx:xx:xx:xx:xx:xx:xx") in release r12 and later. A TCP/IP PING command on this value should be successful.

cciname

Specifies the name that CAICCI will use for this node. This value can be the same as the hostname, but it does not have to be. The name must match the CAICCI name defined at the remote node. For z/OS, the CAICCI name is defined by the SYSID parameter in the ENFPARMS.

STARTUP

If specified, indicates that the connection is attempted as soon as CAICCI starts. This is recommended.

PORT

Specifies the TCP/IP port number (*nnnn*) to use. The default on z/OS with CAICCI 2.1 is 1721 and can be changed on the PROTOCOL(TCPSSLGW...) statement. (The default port for non z/OS machines is 1721.)

RETRY

Specifies the time (*x*) to wait to reconnect if the connection attempt fails or is broken.

CAICCI must be recycled to pick up changes to the ccirmttd file. The ccirmttd file also contains the LOCAL statement, which identifies this machine in a CAICCI network. Example LOCAL statement:

```
LOCAL = BARNA03SRVR BARNA03XPS 32768 startup ALIAS=BARNA03W
```

An ALIAS keyword is required when the CAICCI name field, the second field on the LOCAL statement, is longer than 8 characters. Most CAICCI applications on the mainframe can only support 8-character CAICCI sysids (including cross platform scheduling). The ALIAS setting will be the 8-character name that the local machine is known as by z/OS CAICCI applications.

z/OS Connections

A TCP/IP Gateway PROTOCOL statement is required for any CAICCI cross-platform scheduling communication. This is required even if the individual node definitions are defined on the non-z/OS platform.

If you need a z/OS system to initiate a connection to a distributed system, you must add one NODE and CONNECT statement as follows to your z/OS CAICCI parameters for each distributed system that you want CAICCI to initiate a connection with.

```
NODE(TCPSSLGW,ip-address:port,retry,cciname)
CONNECT(cciname)
```

Note: Keep in mind that the default is for distributed systems to initiate CAICCI connections with a z/OS system. This tends to work better because smaller UNIX and Windows Server systems tend to have more outages. However, you can set it up so that z/OS also initiates connections. If the z/OS system is IPLed or CAIENF or CAICCI is recycled, a connection to a distributed machine would be reestablished slightly faster when initiated by the z/OS system.

Here are the NODE and CONNECT statement descriptions:

TCPIPGW or TCPSSLGW

Identifies what protocol to use for the connection. This value is required.

ip-address

Specifies the numerical TCP/IP address of the node or the hostname that can be resolved to the TCP/IP address. You can specify an IP address in either IPV4 format (dotted decimal notation: "ddd.ddd.ddd.ddd") or IPV6 format ("xx:xx:xx:xx:xx:xx:xx:xx") in release r12 and later

port

Specifies the TCP/IP port number to use at the specified node. If omitted, the port number for the TCPIPGW or TCPSSLGW protocol on the local machine is used.

retry

Specifies the number of minutes to wait between attempts to establish or reestablish the connection.

Retry time must be specified from 1 to 59 minutes.

cciname

Specifies the name that CAICCI will use for this node. cciname is also referred to as the cci sysid.

Chapter 7: Windows to Mainframe: Common Communications Interface

This section contains the following topics:

[PC-to-Mainframe Client-Server Configuration on Windows](#) (see page 203)

[Install CAICCI-PC](#) (see page 204)

[Configurator](#) (see page 204)

[Configure CAICCI for TCP/IP](#) (see page 205)

[Test the Configuration](#) (see page 210)

[Trace a Communications Problem](#) (see page 210)

PC-to-Mainframe Client-Server Configuration on Windows

The client server configuration is suitable for PC-to-mainframe client server applications. It requires installation of the CAICCI-PC component on the PC. Support is available for both 32-bit and 64-bit client applications.

TCP/IP, with or without SSL Support, is used for PC to mainframe connectivity through CAICCI.

TCP/IP

CAICCI communicates between mainframes and PCs using TCP/IP as the network protocol. TCP/IP provides efficient, high-bandwidth connections and is suitable in environments where achieving top performance or the ability to handle high volume is of primary concern.

CAICCI interfaces to TCP/IP through an interface known as the sockets API. If you are running Windows, Microsoft supplies the API as part of the operating system. All you must do is to ensure that you have enabled TCP/IP as a network protocol on your Windows system.

Note: There are also mainframe requirements for using TCP/IP.

Install CAICCI-PC

The PC-to-mainframe client-server configuration requires CAICCI-PC on the participating PCs. Support for both 32-bit and 64-bit client applications is provided.

To install CAICCI-PC on a PC

1. Remove any previous CAICCI-PC by using the CAINDREG program located in the C:\CA_APPSW directory.
2. Download the CCIPCS32 and/or CCIPCS64 file from the installation media or the FTP site:

Note: For information about downloading the installation files (CCIPCS32 and CCIPCS64), see the *Installation Guide*.

3. Run CCIPCS32.EXE or CCIPCS64.EXE.

The InstallShield Wizard appears.

4. Click Next, and following the instructions.

The wizard asks you to accept the CA License and the OpenSSL License and the Original SSLeay License. By default, the component is installed in the C:\Program Files\CA\SharedComponents\CAICCI-PC directory where CA products expect to find it. You have the option to install into a different folder.

Note: As the last step of the installation, the wizard asks if you want to configure CAICCI-PC. Check the box to do the configuration now. Otherwise you can cancel the CAICCI-PC configuration by clicking Finish. You can later set the communications information and other options by accessing the Configurator (CAICCI-PC Properties dialog) from the Start menu (Start>Programs>CA>CA CAICCI-PC>Launch CCIPC Configurator). However, if you were to cancel at this point, your CA products cannot communicate with a host and would ultimately fail. You should set the TCP/IP information and other options before running any CAICCI applications.

Configurator

The Configurator (CAICCI-PC Properties dialog) lets you set SSL options with CAICCI-PC Version 14.0. You can also select various settings for the TCP/IP protocols. After you specify the settings, they remain in effect until changed.

Note: On a Windows XP Professional, 2003, 2008 or Windows 7 computer the Configurator must be run by an Administrator ID because the Configurator updates the registry.

Configure CAICCI for TCP/IP

You configure CAICCI using the Configurator (CAICCI-PC Properties dialog), which is either already running as part of the installation process, or you can run it through the Start menu.

Note: On a Windows XP Professional, 2003, 2008 or Windows 7 computer, the Configurator must be run by an Administrator ID because the Configurator updates the registry.

To configure CAICCI for TCP/IP on Windows

1. Run the Configurator.
The CAICCI-PC Properties dialog opens.
2. Select the TCP/IP tab.
The page for setting TCP/IP parameters opens.
3. Set the TCP/IP parameters as follows:
 - Identify the mainframe CAICCI through CAICCI Server Identification group of fields
 - Define the ID of the CAICCI on the PC in the System Name field
Note: Choose a unique name or leave the field blank to use a name derived from the machine name.
 - Update the other fields as required.
4. Click Apply.
You have configured CAICCI on your PC to use TCP/IP.
5. Select the SSL tab.
The page for setting SSL parameters opens.
6. Update the SSL parameters as required, and click Apply.
Note: The path to find certificates is C:\CA_APPSW, which is specified in the Path environment variable. You can leave the SSL Path field blank or specify a path other than the default.
You have configured CAICCI on your PC to use SSL.
7. Click OK.
The CAICCI-PC Properties dialog closes.

TCP/IP Tab

The TCP/IP tab lets you set TCP/IP parameters for CAICCI on Windows.

This tab contains the following fields to identify the server:

Name or IP address

Specifies the name or Internet address of the server, which acts as a gateway for CAICCI requests.

IPv6 is supported in CAICCI-PC Version 14.0. You can specify an IP address in either IPv4 format (dotted decimal notation: "ddd.ddd.ddd.ddd") or IPv6 format ("xx:xx:xx:xx:xx:xx:xx:xx"). IPv6 format addresses are supported in the CAICCI Server Identification box. Nodenames for IPv6 systems that are mapped to IPv6 addresses on the DNS server are also supported.

If you specify a logical name, TCP/IP is configured to work with a name server, and the name you enter is defined to the name server.

Port

Specifies the port number that the CAICCI server expects to use.

Default: 1202

The tab contains the following field to identify the client:

System Name

Defines a system name to identify your PC uniquely.

By default, CAICCI uses the name that TCP/IP defines. If the name is longer than eight characters, it uses the machine name to derive a different but unique 8-character name. To use a preferred ID, specify a unique new name in the text box.

Default: Name TCP/IP defines

Limits: Eight characters

The tab contains the following fields for CAICCI timeout intervals:

Reply Wait

Specifies the interval to wait for the initial part of a packet to be received in a socket. A value of 0 tells CAICCI to poll forever.

Default: -1 (largest signed 32-bit integer)

Ready to Receive

Specifies the interval to wait for the remainder of the data to be received, after the initial packet. A value of 0 tells CAICCI to poll forever for the rest of the data.

Default: -1 (largest signed 32-bit integer)

Ready to Send

Specifies the interval to wait for a socket to be available to send data. A value of 0 tells CAICCI to poll 9999 times.

Default: 60

Disable CAICCI timeout values or Disable appl timeout values

Specifies whether to ignore any timeout value passed in the Timeout parameter of CAICCI API requests. The CAICCI-PC timeout value will be used instead. You should check this box only when instructed to do so by CA Support.

SSL Tab

The SSL tab lets you set the Secured Sockets Layer parameters for CAICCI on Windows.

The PC must connect to a mainframe server that supports SSL (CCISL) for the protocol to be active. A PC running SSL-enabled CAICCI code that connects to a server that does not support SSL will either revert to the standard unsecured protocol or has its connection request rejected, depending on the selected SSL encryption option below.

This tab contains the following fields to set SSL encryption options:

Force Secure end-to-end connection

Selecting this option informs CAICCI that end-to-end SSL is required for all CCI requests. There must be a secured link in place from the PC to its receiving application's target host including any intermediate hosts acting as routers to the target host. Since SSL is also required for the PC's connection to its mainframe server, selecting this option also forces on the option for Force secure connection from PC to Host.

Force secure connection from PC to Host

Selecting this option specifies that an SSL connection is required by the PC to its mainframe server. If the server does not support SSL, the connection request fails.

Defer decision to host

Selecting this option defers the decision of establishing a secured SSL connection to the mainframe server. An SSL connection will be established only if the mainframe server requires it.

Disable secure connection on the PC

Selecting this option disables SSL on the PC. If the mainframe server requires a secured SSL connection, the connection request fails.

Note: The PC application can programmatically specify and override the settings of the SSL Tab.

The SSL Tab contains the following fields to locate certificates. End-user SSL certificates are now supported in PKCS#12 format and both user and CA certificates can be stored and accessed from the Windows Certificate Store.

SSL Path

Specifies the name of the directory path where CAICCI-PC searches for certificates unless overridden by one of the fields described below.

Client Certificate

This field specifies the absolute path and name of a file (if the file name starts with a "drive_letter:\") or the relative path and name of a file (relative to SSL Path) containing the Public Key Infrastructure (PKI) private key and certificate that the PC uses to identify itself to the mainframe server. If Client Certificate has a file type of "*.p12", the certificate is assumed to be in PKCS#12 format. Otherwise the certificate is assumed to be in PEM format.

The Client Certificate field can also reference a certificate within the Windows Certificate Store. This reference cannot be by filename but rather is through an entity within the certificate. The following methods can be used to reference a certificate within the Windows Store:

- Specify a character string that is contained within the certificate's Subject Name, enclosed within double quotes. For example, specify: "substring". Multiple certificates can match on a substring. The first match is used. Specify enough of the certificate's Subject Name to guarantee the desired certificate is selected. If multiple certificates have the same Subject Name, then do not use this method.
- Specify the entire hex string of the certificate's Thumbprint, enclosed within < >. For example, specify: <xx xx xx xx... xx>. The thumbprint is unique.

PKI Password

This field specifies the password for Client Certificate that allows CAICCI to use the PKI private key. The password is required when Client Certificate specifies a filename. The password for a certificate residing within the Windows Certificate store is required at the time that the certificate is imported into the store.

CA Certificates

This field specifies the absolute path and name of a file (if the file name starts with a "drive_letter:\") or the relative path and name of a file (relative to SSL Path) containing one or more concatenated Certificate Authority certificates that the PC uses to authenticate certificates received from its server.

The CA Certificates field may also reference a CA certificate within the Windows Certificate Store. This reference cannot be through a filename but rather is through an entity within the CA certificate. The following methods can be used to reference a CA certificate within the Windows Store:

- Specify a character string that is contained within the CA certificate's Subject Name, enclosed within double quotes. For example, specify: "substring". Multiple CA certificates can match on a substring. The first match is used. Specify enough of the CA certificate's Subject Name to guarantee the desired certificate is selected. If multiple CA certificates have the same Subject Name, then do not use this method.
- Specify the entire hex string of the CA certificate's thumbprint, enclosed within < >. For example, specify: <xx xx xx xx... xx>. The thumbprint is unique.

CA Directory

This field specifies the absolute path and name of a directory (if the directory name starts with a "drive_letter:\") or the relative path and name of a directory (relative to SSL Path) containing the Certificate Authority certificate files that the PC uses to authenticate certificates received from its server.

The individual Certificate Authority certificate files are named after their subject name hash value. At startup, SSL first loads certificates from the CA Certificates file. During connection time, if SSL cannot find the required CA certificate, it then checks this directory.

SSL Verify Depth

This field specifies the maximum depth of the certificate verification chain. A value of 1 allows the check of the peer certificate and one Certificate Authority certificate. Higher values allow checks for additional Certificate Authority certificates.

Test the Configuration

After configuring the communications protocol, you can test the configuration to see if it has correctly established contact with the host. Error messages appear if the communication fails. Testing using the Configurator (CAICCI-PC Properties dialog) differentiates between errors related to the communications protocol configuration and errors related to the CA solution applications. You can locate the error precisely and quickly, and modify your protocol, options, or both until the communication is successful.

To test the configuration

1. Run the Configurator.
The CAICCI-PC Properties dialog opens.
2. Click Start on the Test tab.
The configuration test starts. The status and any error messages appear in Test Log.
3. Review the messages, correct the errors, and test the corrected configuration.
4. Click OK when you finish with your testing.
The CAICCI-PC Properties dialog closes.

Trace a Communications Problem

CAICCI lets you generate a trace file that can be used when a communications problem occurs.

To trace a communications problem

1. Run the Configurator (CAICCI-PC Properties dialog).
The CAICCI-PC Properties dialog opens.
2. Select the Trace tab.
The page for enabling the trace opens.
3. Check Enabled, check other options as required including the path where you expect the trace to be found and the file name for the trace, and click Apply.
4. Click OK.
The CAICCI-PC Properties dialog closes, and the trace is enabled.
5. Recreate the problem to produce the trace.
The trace is written to the specified trace file
6. Review the trace file to determine the problem.

Note: After you have finished tracing, disable tracing to maximize performance.

Trace Tab

The Trace tab lets you enable CAICCI tracing on Windows.

This tab contains the following check boxes:

Enabled

Enables CAICCI tracing.

Snap Packets

Puts the actual data packets in the trace file.

Dump SSL

Puts additional SSL messages in the trace file.

This tab contains a text box for specifying the name of the trace file. The file name should use the .trc extension. The default is CCITRACE.TRC in the C:\Program Files\CA\SharedComponents\CAICCI-PC directory

Chapter 8: CA-L-Serv

CA-L-Serv is a started task that provides standard services used by various CA Technologies products including CA Endeavor Software Change Manager, CA Bundl, CA TPX Session Management for z/OS, CA Balancing, and CA MIC Message Sharing.

Note: CA Technologies products using CA-L-Serv are referred to as client applications or clients in this chapter.

The CA-L-Serv standard services include:

- Centralized logging facilities
- Centralized messaging facilities
- VSAM file management
- Cross system communications
- SQL table management

This section contains the following topics:

[CA-L-Serv Operation](#) (see page 214)

[CA-L-Serv Configuration](#) (see page 217)

[Configure the File Server](#) (see page 233)

[Define Managed Files](#) (see page 239)

[Manage the File Server](#) (see page 249)

[Maintain Managed Files](#) (see page 251)

[Use the Communications Server](#) (see page 260)

[Configure XCF Communication](#) (see page 263)

[Configure VTAM Communication](#) (see page 267)

CA-L-Serv Operation

CA-L-Serv services are provided by a general purpose entity called the CA-L-Serv kernel and the three servers activated by the kernel.

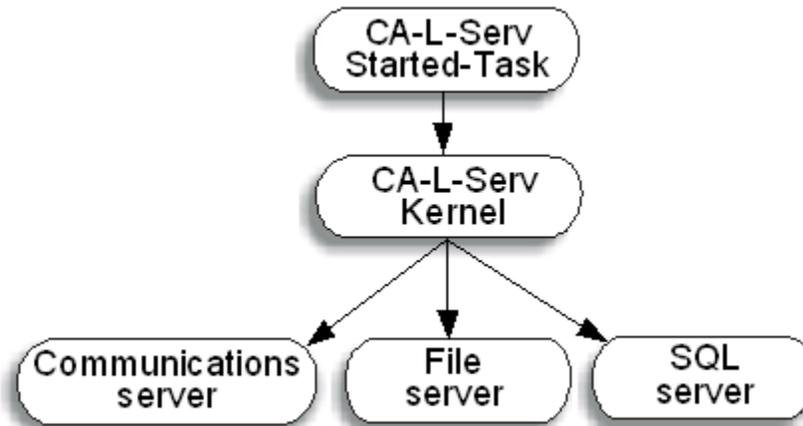
The CA-L-Serv kernel provides three core services for client applications: logging, messaging, and command processing. Through the kernel, you start and stop CA-L-Serv servers, issue commands to those servers, and perform routine maintenance operations, such as file backup and restoration.

The CA-L-Serv servers that run in the CA-L-Serv address space are:

- Communications Server provides cross system communications for client applications.
- File Server manages centralized access to VSAM files used by client applications.
- SQL Server provides SQL access to data stored in relational tables created from VSAM files.

Kernel Server Relationship

The diagram below illustrates the controlling relationship of the kernel and the servers.



The Kernel

The CA-L-Serv kernel provides a number of core services available to client applications that include logging, messaging, and command processing.

Logging

CA-L-Serv has three types of logs for collecting information:

- Message logs, which collect messages about CA-L-Serv activities.
- Trace logs, which collect diagnostic information about CA-L-Serv.
- Change logs, which collect information about updates to files that CA-L-Serv manages on behalf of a client application.

Messaging

Through the CA-L-Serv message service, clients can issue messages to consoles and logs. Additionally, its unique message table structure provides foreign language support and lets you customize messages to the requirement of your site.

Command Processing

Using CA-L-Serv commands and statements, you can:

- Start and stop the servers
- Add or remove files from the management of the file server
- Display information about the current status of the servers
- Perform numerous other administrative procedures

Communications Server

The CA-L-Serv communications server enables a client application running on a z/OS system to communicate with other client applications running on other z/OS systems.

File Server

The CA-L-Serv file server manages access to VSAM files used by client applications.

When you use the file server, only CA-L-Serv accesses the files you place under its management. To read or write to those files, the client application issues I/O requests that are processed by CA-L-Serv.

The files managed by the file server can be organized into file groups. This useful feature includes automatic switching to the next file in the group when the current data set is full (such as in the case of journals) and global file management where an entire group may be backed up, restored, and archived as a single entity.

Because the operating system treats CA-L-Serv as the only user for these managed files, users cannot circumvent the internal security mechanism of a client application to access data. For added security, CA-L-Serv provides a logging feature that you can use to track updates to managed VSAM files.

Note: The external security features introduced with genlevel 9510 are unchanged with this new genlevel of CA-L-Serv. See the *Installation Guide* for a complete description of these features and their implementation.

Sharing VSAM Files in a Multi-System Environment

When used with the communications server, the file server enables client applications to share VSAM files across systems without physically sharing DASD. One system acts as the host system and other systems act as remote systems. The communications server routes I/O requests from remote systems to the host system, where the file server performs all I/O operations against those files.

For performance reasons, CA recommends that processes that involve large numbers of I/O operations should execute on the host system rather than remote systems.

SQL Server

CA-L-Serv includes its own enhanced version of SQL, patterned after ANSI guidelines. The SQL server provides two facilities:

- An internal API relational database provides access to client applications.
- A command interface provides online access to relational databases from a TSO terminal.

Start and Stop CA-L-Serv

The following commands are used to start and stop CA-L-Serv:

- To start CA-L-Serv, issue the following z/OS START command:

```
S task,parm1,parm2,...
```

The task variable represents the name of the CA-L-Serv startup procedure. The parm variables represent one or more parameters from the startup procedure that you want to override.

- To stop CA-L-Serv, issue **one** of the following commands:

```
F task,SHUTDOWN
```

```
P task
```

CA-L-Serv automatically deactivates all servers running in its address space.

Note: Starting CA-L-Serv does not automatically start all of the CA-L-Serv servers. See [Activate the Servers](#) (see page 221) for instructions on defining and activating the servers.

CA-L-Serv Configuration

CA-L-Serv must be configured to identify the system and address space where CA-L-Serv is running.

Choose System and Subsystem Names

The CA-L-Serv "system name" identifies the system where a copy of CA-L-Serv is running. When CA-L-Serv executes on multiple CPUs each CA-L-Serv in the complex identifies each of its partners through the system name specified at startup. By default, if no system name is specified at startup, CA-L-Serv uses the system SMF ID. It is strongly recommended that you use this default value. However, if you need to override the default system name, you can use the SYSNAME= parameter in the CA-L-Serv startup procedure.

The CA-L-Serv "subsystem" name identifies the address space where CA-L-Serv is running. Associated products such as CA Endeavor Software Change Manager use the CA-L-Serv subsystem name to indicate the copy of CA-L-Serv with which they want to interact. When CA-L-Serv executes on multiple CPUs, all CA-L-Serv regions in the complex must share the same subsystem name.

Most sites can use the default subsystem name LSRV. However, if you run multiple copies of CA-L-Serv on any system, assign a unique subsystem name to each copy. To do this, change the value for the SSNM= parameter in the CA-L-Serv startup procedure.

Use Two Subsystem Names

If you run both test and production copies of CA-L-Serv on system SYS01, use the same system name for both copies of CA-L-Serv but use different subsystem names. For example, use LSVT for the test CA-L-Serv and LSVP for the production CA-L-Serv.

Identify CA-L-Serv to Client Applications

Client applications generally reference the CA-L-Serv default subsystem name (LSRV). If a client application uses a different subsystem name, or you change the CA-L-Serv subsystem name, you must reference the new name as follows:

- In batch jobs (such as a CA Bundl collection job), add the following DD statement to the JCL:

```
//SSN$sysname DD DUMMY
```

- For interactive environments (such as a CA Bundl TSO View session), add the following statement to the CLIST:

```
ALLOC F(SSN$ssname) DUMMY REUSE
```

- For clients who identify CA-L-Serv through a parameter, specify the CA-L-Serv subsystem name on that parameter. For example, specify the CA-L-Serv subsystem name on the CA MIC Message Sharing ISSNAME parameter.

Note: See the documentation for the client application for additional information on default subsystem names and overriding them.

Refer to Two Different Subsystem Names

Suppose that you are running two copies of CA-L-Serv (one for test files, the other for production files) on the same system. To ensure that jobs use the appropriate copy of CA-L-Serv, add DD statements to the jobs' JCL.

Provide Operating Values for CA-L-Serv

Operating values are provided to CA-L-Serv in members of the CAI.CAW0OPTN data set, which was created when you installed CA-L-Serv. The data set is allocated to the CA-L-Serv startup procedure under the LDMCMND DD statement.

Members that Provide Operating Values

The CAW0OPTN data set contains three categories of members that define operating values for CA-L-Serv:

- Message tables (LDMMSGGS) points to message tables containing message definitions for CA-L-Serv or its client.
- Command members (LDMPARM) lists all commands and command members that are executed when CA-L-Serv starts up.
- SQL Definition Members (LDMSQDEF) defines SQL relational tables. This member is used only by the CA-L-Serv SQL server.

Note: For greater flexibility, the LDMPARM member is specified in the CA-L-Serv startup procedure and it is possible to direct CA-L-Serv to execute different sets of startup commands for different executions. On the other hand, the other two members' names are fixed and they must not be changed.

These three categories of CAW0OPTN members are further described in the following sections.

Define Message Tables for CA-L-Serv

A message table contains definitions of the messages that are issued by CA-L-Serv or one of its client applications. The definitions include the message number, message text, symbolic variables, and standard z/OS routing and descriptor codes.

The CAW0OPTN member LSERVMSG, which is downloaded at installation time, contains the definitions of messages necessary for CA-L-Serv processing.

CA-L-Serv may use additional message tables to issue messages on behalf of any client application that sends a message request to CA-L-Serv. See the installation and configuration documentation for the associated product for specific information concerning its implementation and use of CA-L-Serv messaging services.

Note: The message table for an associated product can be copied either to the CA-L-Serv CAW0OPTN data set or to any data set included in the concatenation for the //LDMCMND DD statement in the CA-L-Serv startup procedure.

How to Tell CA-L-Serv which Message Tables to Use

The LDMMSGGS member of the CA-L-Serv CAW0OPTN data set points to all message tables used by CA-L-Serv.

The sample LDMMSGGS member in the CA-L-Serv CAW0OPTN data set contains one INCLUDE statement for the LSERVMSG message table. Additional message tables must be defined in LDMMSGGS

It is recommended that you provide a separate message table for CA-L-Serv and additional message tables for each client. For example, you might provide message tables for CA-L-Serv, CA Bundl, and CA Endeavor Software Change Manager as shown here:

LDMMSG **This member points to all message tables used by CA-L-Serv.**	
INCLUDE LSERVMSG	**This member defines CA-L-Serv messages.
INCLUDE BNDLMSG	**This member defines CA Bundl messages.
INCLUDE NDVRMSG	**This member defines CA Endeavor Software Change Manager messages.

Customize Messages

The message text contained within any message table provided by CA Technologies can be reworded to suit your needs; this may include translation to languages other than English. For information on altering message tables, see the *Message Reference Guide*.

If you need to customize a message table, do not modify the original member. Copy the original message table into a new member and modify the corresponding INCLUDE statement in LDMMSG.

Provide Non-English Messages

If you want to provide message tables in a language other than English, create additional message tables for each client application. Because most CA Technologies products are shipped only with the English version of the message table, you may have to translate message definitions yourself.

For example, you might provide these members if you have English and French versions of CA-L-Serv and CA Bundl messages:

LDMMSG **This member points to all message tables used by CA-L-Serv.**	
INCLUDE LSENGMSG	**Defines CA-L-Serv messages in English.
INCLUDE LSFRNMSG	**Defines CA-L-Serv messages in French.
INCLUDE BNENGMMSG	**Defines CA Bundl messages in English.
INCLUDE BNFRNMSG	**Defines CA Bundl messages in French.

For additional information on providing language support, see the *Message Reference Guide*.

Activate the Servers

Starting CA-L-Serv activates only the core services referred to as the CA-L-Serv kernel. Depending on the client application that will use CA-L-Serv services, you will need to activate one or more servers. A convenient way of doing this is with command members.

Command Members

A command member is a member of the CA-L-Serv CAW0OPTN data set that contains CA-L-Serv commands. You can use command members to start the CA-L-Serv servers and to perform CA-L-Serv tasks on behalf of a client application (such as placing files under the management of CA-L-Serv) whenever you start the CA-L-Serv kernel.

A sample command member called LSVPARM is provided in the CA-L-Serv CAW0OPTN data set to illustrate the layout of a typical CA-L-Serv startup command member.

Note: The CA-L-Serv command members cannot contain z/OS commands or commands for the CA-L-Serv clients. They can only contain CA-L-Serv commands.

Store CA-L-Serv Command Members

Command members can be stored in the CA-L-Serv CAW0OPTN data set or in any data set included in the concatenation for the //LDMCMND DD statement in the CA-L-Serv startup procedure.

Start CA-L-Serv Servers

All startup commands for CA-L-Serv and its servers should be included in the LSVPARM command member. Within this command member, you can provide commands to start and set operational values for CA-L-Serv, the file server, the communications server, and the SQL server.

The ATTACH commands get placed in the LSVPARM member. You would also include commands that define the environment of each CA-L-Serv server. For example, you would specify a LOGID with each ATTACH command to define the log used for the messages of each server. See the LSVPARM member in the CAW0OPTN data set for examples of startup commands.

Provide Command Members for CA-L-Serv Clients

You should provide a command member for each of the CA-L-Serv client applications. In each of these members, specify the commands that CA-L-Serv should issue upon startup to enable the operations for that client. The ADDPOOL, ADDFILE, and ADDLOG commands get placed in these members.

For example, a BNDLPARM member could provide buffer pools for CA Bundl files, place those files under the management of CA-L-Serv, and associate change logs with each of those files. Another member called NDVRPARM, for example, could perform similar tasks for the CA Endeavor Software Change Manager journals; the member LSVPARM may contain startup commands for CA-L-Serv servers, and LDMPARM contain INCLUDE statements for LSVPARM, BNDLPARM, and NDVRPARM.

Point to Startup Members

The LDMPARM member of the CA-L-Serv CAW0OPTN data set lists all command members that are to be executed when CA-L-Serv is started up. In this member, specify an INCLUDE statement for each command member that CA-L-Serv should use at startup time.

To include the command member that activates the CA-L-Serv servers and the command members that execute CA-L-Serv commands on behalf of CA Bundl and CA Endeavor Software Change Manager, add the following entries in LDMPARM:

LDMPARM **This member contains the list of command members which are to be executed when CA-L-Serv is started.**	
INCLUDE LSVPARM	**This member starts up CA-L-Serv servers.
INCLUDE BNDLPARM	**This member contains CA-L-Serv tasks for CA Bundl.
INCLUDE NDVRPARM	**This member contains CA-L-Serv tasks for CA Endeavor Software Change Manager.

The commands in each member, and the members themselves, are read and executed in the order in which they appear.

Define SQL Relational Tables for CA-L-Serv

An SQL *definition member* is a parameter data set member containing definitions of SQL relational tables used by the SQL Server.

SQL table definition members can be stored in the CA-L-Serv CAW0OPTN data set or in any data set included in the concatenation for the //LDMCMND DD statement in the CA-L-Serv startup procedure.

Tell CA-L-Serv which SQL Definition Members to Use

The LDMSQDEF member of the CA-L-Serv parameter data set points to all SQL definition members. All SQL definition members must be included in the LDMSQDEF member.

For example, to point to the member PRODSQL containing SQL table definitions, you would include the PRODSQL member as shown here:

LDMSQDEF **This member points to all SQL definition members used by CA-L-Serv**	
INCLUDE PRODSQL	**This member defines SQL tables for an unspecified client application.

Note: The LDMSQDEF member is read automatically at startup by the SQL server. Consequently, the name of the member must not be altered.

Multiple System Environment Configuration

If you are running the file server and the communications server in a multiple system configuration, you can share the CA-L-Serv parameter data set among systems. This would mean that multiple systems share the physical DASD upon which the CA-L-Serv parameter data set is stored. With such a configuration, you can centralize control and more easily maintain CA-L-Serv.

Use Separate Startup Members for Host and Remote Systems

If you are sharing the CA-L-Serv parameter data set among systems, you can provide separate startup command members for each system: one for the system where the host file server is running and one for each system where a remote server is running. These members should be referenced using INCLUDE statements in the LDMPARM member of the CAW0OPTN data set. (Host and remote servers are described in the File Server section.)

Having these separate startup command members allows you to easily distinguish commands for the host server from commands for remote servers. In addition, if the host server goes down, you can change host servers without stopping CA-L-Serv. See Configuring the file server for an example of how you can implement new host and remote servers using command members, without shutting down CA-L-Serv.

For example, you might provide the HOSTPARM member to start CA-L-Serv servers and establish the host server, and REMOPARM member to start CA-L-Serv servers and establish the remote server.

Share a Startup Member between Host and Remote Systems

The IFSYS, ENDIF and ELSE statements may be used to direct CA-L-Serv to execute commands only on certain systems. The IFSYS statement identifies the systems where the ensuing block of commands will be executed. An ELSE or ENDIF statement ends the block.

On an IFSYS statement, you need to specify the SMF ID of the system where the following block of commands will be executed. You can specify one or more system names on each IFSYS statement.

This feature can be used to manage several CA-L-Serv programs executing on multiple systems with a single LSVPARM member.

See the *Reference Guide* for detailed information about the IFSYS, ELSE and ENDIF statements.

Utilization of IFSYS/ENDIF Statements

Suppose that you want to use IFSYS/ENDIF blocks to initiate the following command members:

- HOSTPARM (for CA-L-Serv startup commands on system SYS01, where the host file server resides)
- BNDLPARM (for CA Bundl file management commands on the host system)
- NDVRPARM (for CA Endeavor Software Change Manager file management commands on the host system)
- REMOPARM (for CA-L-Serv startup commands on systems SYS02 and SYS03)

You can specify the following statements in the LDMPARM member:

```
IFSYS  SYS01
      INCLUDE  HOSTPARM
      INCLUDE  BNDLPARM
      INCLUDE  NDVRPARM
ENDIF
IFSYS  SYS02, SYS03
      INCLUDE  REMOPARM
ENDIF
```

Use the ELSE Statement

You can also use the ELSE statement within an IFSYS/ENDIF block. The ELSE statement will execute commands on all systems that do not match the system names specified with the IFSYS statement. For example, if your complex has only the three systems named above, you can achieve the same result with the following statements:

```
IFSYS  SYS01
      INCLUDE  HOSTPARM
      INCLUDE  BNDLPARM
      INCLUDE  NDVRPARM
ELSE
      INCLUDE  REMOPARM
ENDIF
```

Deactivate CA-L-Serv Servers and Tasks

You can also provide command members that stop the CA-L-Serv servers or CA-L-Serv file management tasks. These members are useful if you want to stop CA-L-Serv servers without shutting down the CA-L-Serv kernel.

A command member that would stop CA-L-Serv servers would typically contain DETACH commands for the servers being deactivated. You can issue the commands in these command members by using the READ command, which is described in detail in the *Reference Guide*.

Generally the READ command can be used to execute any lengthy sequence of elementary commands (such as file allocation and de allocation) especially when these commands will be executed on a routine basis.

Note: In the LDMPARM member of the CA-L-Serv parameter data set, do not include command members that stop servers. This would cause the specified servers to be stopped right after they are activated.

Establish New Host and Remote Servers

Suppose that you want to shut down a local file server in order to start a multiple system setup with host and remote servers and that the specified command members exist in the CA-L-Serv parameter data set.

To switch to the new host/remote servers, you would:

- Stop CA-L-Serv servers for the local system:

READ STOPLSRV (on the local system)

- Start CA-L-Serv servers for the new host system:

READ HOSTPARM (on the new host system)

- Start CA-L-Serv servers for each new remote system:

READ REMOPARM (on each new remote system)

Collect Data in the CA-L-Serv Logs

A log is a set of sequential files that collects information about CA-L-Serv. Three types of logs are provided:

- Message logs collect messages about commands issued to CA-L-Serv and its responses to those commands. They also collect CA-L-Serv messages that indicate potential problems or significant events.
- Trace logs collect diagnostic information about CA-L-Serv. This information is used by CA Technologies Support.
- Change logs collect information about updates to files that CA-L-Serv is managing. Change logs contain the following types of information:
 - The ID of the user making an update.
 - The type of operation: add, delete, or replace.
 - The dname of the file being updated.
 - The record key (shown in character and hexadecimal formats).

Default Logs Provided by CA-L-Serv

By default, CA-L-Serv provides two types of logs:

- A message log named MSGLOG, defined as a class A SYSOUT data set. CA-L-Serv automatically directs messages to this log unless you remove the log (through a REMOVELOG command) or you provide a different message log.
- A trace log named TRACE, defined as a class A SYSOUT data set. Although CA-L-Serv does not automatically collect trace messages, you should not remove the trace log - keep it available just in case you need it.

Because CA-L-Serv provides you with these default logs, you do not need to define them unless you want to use a cataloged data set for them or you want them directed to a different SYSOUT class.

CA-L-Serv does not provide change logs or auxiliary message logs by default. You need to define them using the ADDLOG commands.

Characteristics of CA-L-Serv Logs

All CA-L-Serv logs have the following generic characteristics:

- Each log can contain up to four cataloged data sets or a single SYSOUT data set. When more than one cataloged data set is used they must share the same logical record length (LRECL).
- CA-L-Serv automatically switches to a new file when the current file becomes full. However, CA-L-Serv overwrites data when all files become full.
- Each system must have its own log files. You cannot share log files between copies of CA-L-Serv. If you try to share log files, CA-L-Serv terminates during initialization.

Additional Considerations for Logs

You should keep the following in mind when setting up your logs:

- Log files cannot be archived through CA-L-Serv. However, you can view their contents on line through CA SYSVIEW Performance Management or a similar mechanism.
- The order in which you define log files to CA-L-Serv determines which file CA-L-Serv uses first.
- After defining change logs, you must assign them to managed files. See [Assigning a Change Log to a File](#) (see page 241).
- CA-L-Serv does not support multi-volume log datasets. Make sure that each log dataset is allocated on a single volume. If multiple datasets are allocated for a log, then they may each be placed on a different volume if desired.

Define Log Files

To define a log file, you need to issue an ADDLOG logname command. Provide log files by including one of these parameters on your command:

Type of Log File	ADDLOG Operand	Considerations
Cataloged data set	DSNAMES(<i>dsnames</i>) Specify up to four data set names, delimited by commas or spaces.	If all data set names cannot fit into one single LSVPARM line, use commas as continuation characters for the ADDLOG command.
SYSOUT data set	SYSOUT(<i>class</i>) Specify a single SYSOUT class.	Use SYSOUT data sets if you want to print log files.
Data sets referenced on DD statements in the CA-L-Serv startup procedure	DDNAMES(<i>ddnames</i>) Specify up to four DD names, delimited by commas or spaces.	For each DD name, add a DD statement to the CA-L-Serv startup procedure.

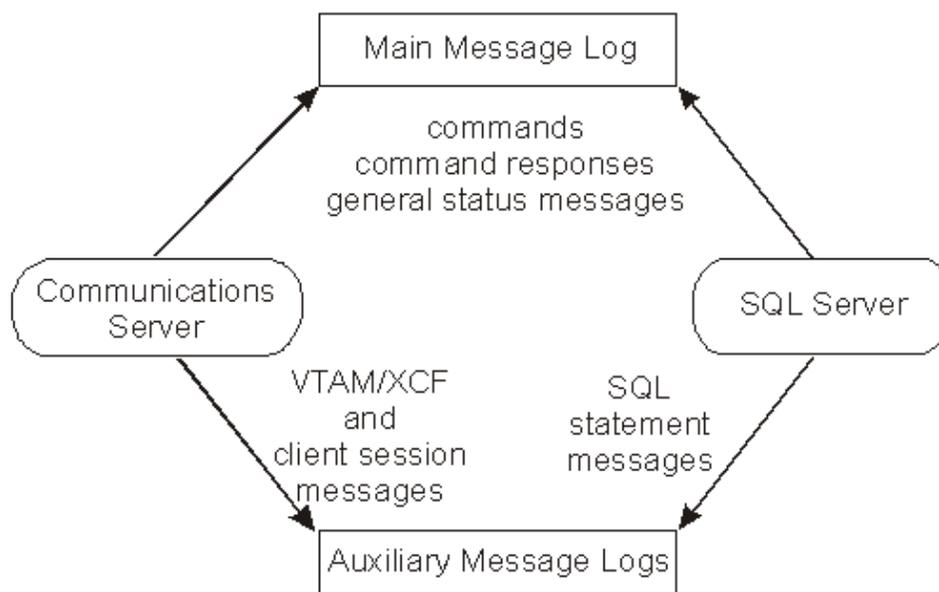
For example, to use data sets MSGLOG.DATA.SET.ONE and MSGLOG.DATA.SET.TWO as message log files, specify this command:

```
ADDLOG MSGLOG DSNAMES(MSGLOG.DATA.SET.ONE ,
                      MSGLOG.DATA.SET.TWO)
```

Auxiliary Message Logs

The CA-L-Serv communications server and SQL server issue a variety of messages. Because you may want to isolate general purpose messages from the more specialized messages that these servers can issue, CA-L-Serv provides an auxiliary log capability.

When you use auxiliary logs, you can direct messages from the communications and SQL servers as shown in the following illustration.



Provide Auxiliary Logs

To provide auxiliary logs for the communications server and the SQL server, take the following actions:

1. Define log files through an ADDLOG command.
2. Associate the log with the appropriate CA-L-Serv server through the LOGID parameter on the ATTACH command. For example, you can start the communications server and associate it with the auxiliary log LCOMLOG through this command:
`ATTACH COMMSERVER LOGID(LCOMLOG)`
3. When you provide an auxiliary log for the SQL server, indicate what type of messages CA-L-Serv should log. By default, CA-L-Serv logs only error messages about SQL statements that it has read. To log all SQL statement messages, including messages about every SQL statement that CA-L-Serv has read, specify `AUDIT(STATEMENTS)` on the ATTACH command.

Log File Operations

After you define a log, CA-L-Serv automatically opens and starts recording information in it. You can issue a command to stop writing to a log at any time. CA-L-Serv also provides commands for reopening a closed log or permanently closing a log. The following table lists the commands that CA-L-Serv provides for opening, closing, and checkpointing logs.

OPENLOG *logname*

Opens a log and starts recording in it.

CLOSELOG *logname*

Stops recording to the log and temporarily closes it.

WRITELOG *logname*

Writes all active buffers to the log to ensure that the most recent events have been recorded.

REMOVELOG *logname*

Stops recording to the log and permanently closes it. (The log must be redefined using the ADDLOG command before CA-L-Serv can use it again.)

SWITCHLOG *logname*

Closes the current log and opens the next log in the list. (The list is defined on the ADDLOG command.)

For more information about these commands, see the *Reference Guide*.

Print Logs

If you define a log as a SYSOUT file, you can use the PRINTLOG command to print the contents of the file and open a new SYSOUT log. To print the log, issue the following command from a console:

```
PRINTLOG Logname
```

CA-L-Serv sends the log to the SYSOUT class that is defined on the ADDLOG command or on the DD statement for the log.

Note: The PRINTLOG command provides a convenient means of directing the contents of a CA-L-Serv log to an output sysout class where it can be archived or purged without having to recycle CA-L-Serv.

Obtain Dumps

If CA-L-Serv experiences an error or abend, you can obtain the following types of dumps for diagnostic purposes:

- An unformatted SVC dump of the CA-L-Serv address space and selected areas of common storage. This dump is placed in a SYS1.DUMPxx data set.
- A formatted dump of the CA-L-Serv address space. This dump is placed in the data set identified by a //SYSUDUMP or //SYSABEND DD statement in the CA-L-Serv startup procedure.

By default, CA-L-Serv generates an unformatted SVC dump.

You can disable the SVC dump feature of CA-L-Serv by issuing the following command:

```
OPTIONS SVCDUMP(NO)
```

Note: The CA-L-Serv recovery routines suppress the generation of duplicate SVC dumps. CA recommends that the SVCDUMP feature be left active at all times, particularly on complex production environments where formatted dumps are often not sufficient to investigate the causes of problems.

Display Information about CA-L-Serv

To display information about CA-L-Serv, issue a DISPLAY command. Specify one or more of these values on the command:

ACTIVE

Displays a list of active CA-L-Serv servers. (See message LDM0420I.)

INIT

Displays values that cannot be changed while CA-L-Serv is running. Values for the CA-L-Serv kernel include the name of the member that points to startup command members; whether CA-L-Serv reuses intercepts; the CA-L-Serv subsystem name; and system IDs. (See message LDM0403I in the *Message Reference Guide*.)

LOGS

Displays status information about the CA-L-Serv message, trace, and change logs. Each log file is listed separately. (See message LDM0750I.)

MSGTABLE

Displays a list of message tables, including what language each table is in, and how many messages each table contains. (See message LDM0422I.)

OPTIONS

Displays values that you can change while CA-L-Serv is running. Values for the CA-L-Serv kernel include your SVC dump option. (See message LDM0404I.)

SSNAME

Displays a list of subsystems (including CA-L-Serv) that are running on a system. Status information for each subsystem is also shown. (See message LDM0410I.)

STORAGE

Displays information about storage use, including the size of a storage block, how many blocks are allocated, and how many blocks are used. (See message LDM0425I.)

VERSION

Displays the CA-L-Serv release and maintenance level. (See message LDM0402I.)

For example, to display lists of subsystems and CA-L-Serv servers, issue this command:

```
DISPLAY SSNAME ACTIVE
```

Note: You can also use the DISPLAY ALL format of the command to generate a complete report of the current status of CA-L-Serv. See the *Reference Guide* for full details on the DISPLAY command.

Configure the File Server

The CA-L-Serv file server manages access to VSAM files used by client applications. To access a file, the client issues an I/O request to the file server, and the server accesses the file on behalf of the client. Only the file server has direct access to the files that you place under its management.

Important! You must run the file server when you run certain client applications (such as CA Bundl) and also if you want to use the CA-L-Serv SQL server.

Start and Stop the File Server

To automatically start a file server when you start a CA-L-Serv started task, specify the ATTACH FILESERVER command in the startup command member. The format of the ATTACH FILESERVER command varies based on the following considerations:

- If you have a multiple system environment, you must define one file server as the host and all others as remote.
- If you have a multiple system environment, do the CA-L-Serv started tasks that run each file server also run a communication server? If a file server uses a communication server executing in another CA-L-Serv region, you must identify this communication server to the file server.
- If you need to adjust the data buffer size of a file server, you must define the new size.

Command-Members Commands

CA Technologies recommends the use of command members to place files under CA-L-Serv management and also to issue commands on a routine basis (such as commands which allocate or deallocate data sets). If you use command members to issue the commands associated with the tasks in the following sections, you will need to set up these commands only once. Thereafter, they are issued automatically. See the CA-L-Serv section for information on using command members.

Console-Issued Commands

You can issue any of the commands listed in this chapter from a console after the file server is started. However, the settings and definitions associated with console issued commands are lost when the file server is stopped, and you must reissue these commands when the file server is restarted.

Start Command

In this example of the file server start command, the file server is the host server in a multiple system environment. The file server uses a separate started task for communications (the subsystem name of the communications server is LSVP), and the data buffer size of the file server is set to 32 kilobytes.

```
ATTACH FILESERVER SERVERTYPE(HOST) COMMSERVERSSN(LSVP)
      BUFFERSIZE(32768)
```

See the *Reference Guide* for additional information on the ATTACH command.

Stop Command

To stop the file server, specify the following command from a terminal or in a command member that issues shutdown commands:

```
DETACH FILESERVER
```

To issue the DETACH command from a command member, specify the following command:

```
READ member
```

member

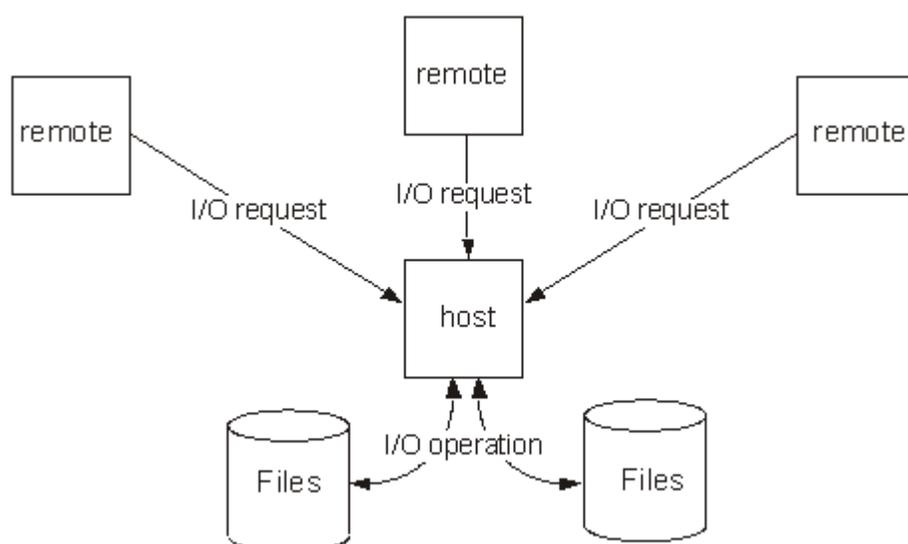
Contains the necessary DETACH command.

Define Host and Remote Servers

In a multiple system environment, you must designate host and remote file servers. Each CA-L-Serv configuration has only one host file server; all other file servers are remote servers.

You should run the host server on the system where the most file I/O requests will be issued. Because the host server performs all I/O operations against a set of managed files, all managed files must be accessible from the system running the host file server.

Remote servers do not perform I/O operations on files. Their I/O requests are forwarded to the host server, as shown in the following graphic:



Designate Servers as Host or Remote

To designate host and remote servers, do the following:

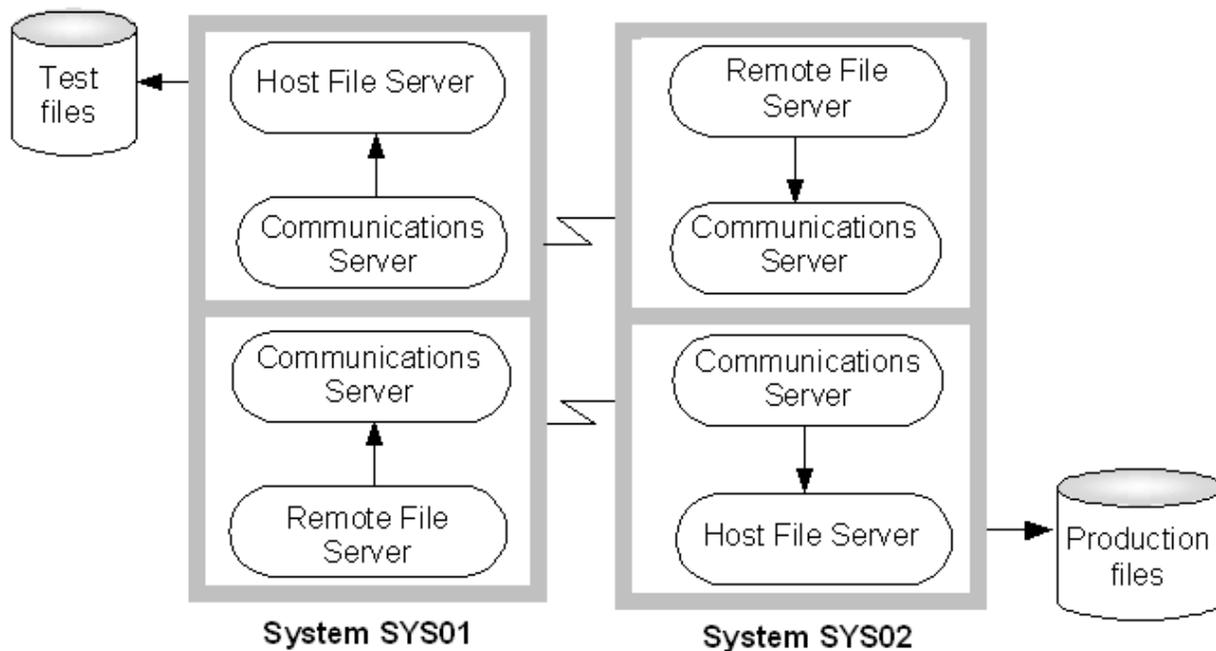
- On the host system, specify this command in the appropriate command member:
`ATTACH FILESERVER SERVERTYPE(HOST)`
- On each remote system, specify this command in the appropriate command member:
`ATTACH FILESERVER SERVERTYPE(REMOTE)`

Note: You do not have to start your host server before your remote servers. Until all servers are started, you will receive messages as the servers try to contact each other.

Configure Communications in a Multiple-System Environment

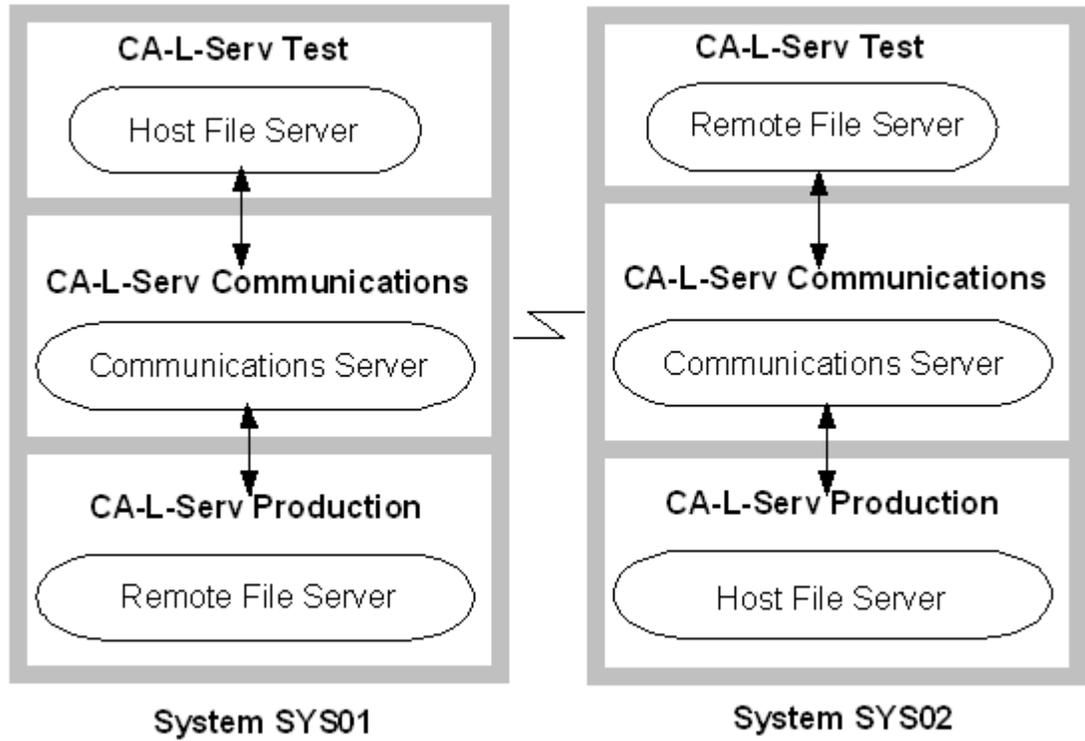
In a multiple-system environment, you will use the CA-L-Serv communications server to enable client applications running on remote systems to access the files managed by the host system. The file server and the communications server must run in each copy of CA-L-Serv.

A typical implementation involving test and production copies of CA-L-Serv running on two Z/OS images is represented by the following graphic:



In the previous example, each copy of CA-L-Serv runs its own communications server. However, a specialized CA-L-Serv started task can provide communications services for multiple copies of CA-L-Serv.

The following graphic illustrates this scenario:



Identify the Started Task for Communications

When the CA-L-Serv started task that runs the file server also runs the communications server, the file server communicates with other systems through the communications server in its own address space.

When a CA-L-Serv started task does not provide its own communications services, you must identify the started task that provides these services to the file server. You can do this by including extra information on the ATTACH command that starts the file server, as shown here:

```
ATTACH FILESERVER SERVERTYPE(type) COMMSERVERSSN(ssname)
```

The COMMSERVERSSN(*ssname*) parameter provides the subsystem name for the CA-L-Serv started task that provides communications services.

For example, suppose you are running a test copy and a production copy of the file server and you want both copies to use the communications server running in the LSVP address space.

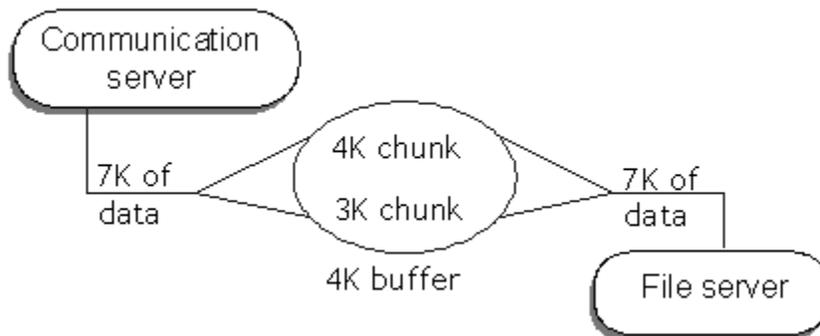
To do this, issue the following command to the test copy of CA-L-Serv:

```
ATTACH FILESERVER SERVERTYPE(type) COMMSERVERSSN(LSVP)
```

Sharing a Communications Server between a number of CA-L-Serv regions is particularly useful to limit the number of VTAM definitions or XCF groups in large complexes where many systems run multiple copies of CA-L-Serv.

Adjust the Size of Data Buffers

CA-L-Serv uses a buffer to temporarily store data that the file server is receiving from the communications server. When the file server issues a request to receive data, CA-L-Serv moves data from the communications server into this buffer. If the buffer is not large enough to accommodate the data, CA-L-Serv splits the data into appropriately sized chunks, moves the data, and then reassembles it, as shown here:



Determine Buffer Size

The default size of the data buffer is 4096 bytes. To determine if you need to change the default value, issue the `DISPLAY SYSTEMS` command and compare the values under the `RECV` and `SPLITS` columns. If `SPLITS` is high relative to `RECV`, you may want to define a larger buffer to improve performance.

The size of the data buffer does not have to match the transmission buffers size of the communications server. However, because the size of the data buffer has an impact on both virtual storage use and performance, you should try to find an optimal size. If the buffer is too large, virtual storage space is wasted. If the buffer is too small, performance suffers from the additional overhead of splitting and reassembling data.

Adjust Buffer Size

To change the buffer size, adjust the value for the `BUFFERSIZE` parameter on the `ATTACH` command that starts the file server, as shown below:

```
ATTACH FILESERVER BUFFERSIZE(bytes)
```

Define Managed Files

There are several setup tasks associated with defining CA-L-Serv managed files.

The tasks and useful notes are listed in the following table:

Task	Notes
Place files under the management of the file server.	Mandatory.
Define LSR buffer pools.	Optional. Check the documentation of the client application to determine which files will be assigned to private pools (NSR) and which files will be assigned to shared pools (LSR).
Set file access parameter.	Optional. You can generally use the default values.
Tune I/O performance.	Optional. You can generally use the default values.
Propagate the ENQ requests across systems.	Mandatory if you are running the file server on multiple systems (host and remote file servers).

Task	Notes
Protect managed data sets using external security (CA Top Secret, CA ACF2, RACF)	Mandatory. For more information, see the <i>Installation Guide</i> .

Place Files under the Management of CA-L-Serv

Consider the following before placing a file under CA-L-Serv management:

- Before it can be opened, a VSAM file must be connected to a buffer pool. By default, VSAM assigns each file to a private (NSR) buffer pool. Optionally, you can direct the file server to create shared (LSR) buffer pools and assign your files to these buffer pools.
- Deciding whether to use NSR (non shared) or LSR (locally shared) buffer pools for a given VSAM data set is the function of a number of criteria including, the mix of I/O operations performed by the client application. The application documentation will generally recommend which of its files should be assigned NSR and LSR pools and provide sample definitions.
- You can assign a change log to each file. A change log collects information about updates to a file. See CA-L-Serv Configuration for more information on change logs.
- You can assign each file to a file group. The files in a file group contain data used by a particular client. All of the files in a file group must have the same file group characteristics. The client application documentation will specify which of its files must be part of a file group.

ADDFILE Command

Use the ADDFILE command to place a file under the management of CA-L-Serv. On the ADDFILE command, specify the ddname. If you did not allocate the file in the CA-L-Serv startup procedure, specify the dsname also.

Note: In a multiple system environment, issue ADDFILE commands only on the system where the host server is running.

Assign a File to a Pool

To assign a file to an LSR buffer pool, specify the POOL parameter with the ADDFILE command, where *nn* is the number for the pool. You must define LSR buffer pools using the ADDPOOL command prior to assigning files to them.

```
ADDFILE ddname POOL(nn)
```

Assign a Change Log to a File

To assign a change log to a file, specify the LOGID parameter with the ADDFILE command, where *logname* is the name of the change log. You must define the change log to CA-L-Serv using the ADDLOG command prior to assigning the log to a file.

```
ADDFILE ddname LOGID(logname)
```

Assign a File to a File Group

To assign a file to a file group, specify the GROUP parameter with the ADDFILE command, where *id* is the group ID of the file group. If the file group does not exist, it is automatically defined.

```
ADDFILE ddname GROUP(id)
```

For example, to place the CA Bundl file DPMFSIF (dsname BUNDL.V47GA.DPMFSIF) under the management of CA-L-Serv, assign the file to LSR buffer pool 3, assign it a change log named BNDCHG3, and assign it to file group BNDGRP3, issue the following command:

```
ADDFILE DPMFSIF BUNDL.V47GA.DPMFSIF GROUP(BNDGRP3)  
LOGID(BNDCHG3) POOL(3)
```

Note: Pool 3 and change log BNDCHG3 must be defined prior to issuing the above command. Group BNDGRP3 is automatically defined by the above command.

Buffer Pools

VSAM uses buffer pools to read and write records from disk and to transfer records to and from a user provided area in the client address space. There are two types of buffer pools:

- *Private buffer pools*, which are reserved for a single file.
By default, VSAM creates a private buffer pool when the data set is initially opened. You only need to define private buffer pools to CA-L-Serv if you wish to override the VSAM defaults, which can be done by using the BUFNI, BUFND, and STRNO keywords of the ADDFILE command.
- *Local shared resource (LSR) buffer pools*, which are shared by a set of files you designate. Files that share a buffer pool should have similar buffer characteristics. CA Technologies recommends that you use LSR buffer pools for files with similar characteristics because this will improve service times for most types of I/O processing.

You can define up to 15 LSR buffer pools using the ADDPOOL command.

See the documentation for the client application for recommendations on which data sets should be assigned to private (NSR) or shared pools (LSR).

Determine the Size and Number of Buffers

When you define LSR buffer pools, you must set the number of buffers per pool and the size of each buffer. Follow these guidelines to determine your number and size requirements:

Number of buffers

The number of buffers can affect system performance. If you define too few buffers of a given size, delays can occur in servicing I/O requests. If you define too many buffers, virtual storage is wasted.

Buffer size

The buffer size should correspond to the control interval (CI) size of the file. When VSAM reads a record, it needs a buffer that is at least as large as the CI size. Because VSAM files can have different buffer characteristics, buffer pools must be large enough to accommodate the CI size of the data and index portions of files without being wasteful.

Note: For more information about NSR and LSR pools as well as the IBM recommendations on determining an adequate size for your buffer pools, see the IBM publication *DFSMS: Using Data Sets*.

Define LSR Buffer Pools

To define an LSR buffer pool, in the CA-L-Serv parameter data set, specify:

```
ADDPPOOL nn (size,count)... STRNO(nnn)
```

nn

Specifies a number to the pool with *nn*. Specify a value from 1 to 15.

size

Specifies a buffer size in bytes. Specify one to four (*size,count*) pairs.

count

Specifies the number of buffers that CA-L-Serv should allocate in that size (the minimum number is 3).

nnn

Optional. Specifies the *nnn* number of VSAM strings assigned to the buffer pool. The STRNO count specifies the maximum number of concurrent I/O requests against the buffer pool that VSAM will be able to process. Specify a value from 3 to 255 (the default is 16).

Note: In multiple system environments, issue ADDPOOL commands only on the system where the host server is running.

For example, to allocate buffer pool 3 with eight 1,024 byte buffers and four 32,768 byte buffers, specify this command in the CA-L-Serv parameter data set:

```
ADDPPOOL 3 (1024,8) (32768,4)
```

Evaluate Buffer Pool Usage

To see how well your buffer pools are being used, issue the DISPLAY commands described below.

- Issue the DISPLAY BUFFERPOOL command:

```
DISPLAY BUFFERPOOL
```

In the output, a high number of BFRFND (buffer find) counts relative to BUFRDS (buffer read) counts are desirable for each buffer size.

- Issue the DISPLAY STATISTICS=SERVICE command:

```
DISPLAY STATISTICS=SERVICE
```

In the output, the following service times are desirable:

- Service times below 40 milliseconds for PUT (write) and DIRGET (direct read) requests.
- Service times below 10 milliseconds for SEQ GET (sequential read) requests.

Note: The BFRFND, BUFRDS, UIW and NUIW keywords that appear on these CA-L-Serv reports are VSAM terminology. For a full discussion of these terms, see the SMS manual Using Data Sets together with the IBM recommendations on their interpretation.

Override VSAM Defaults for Private Buffer Pools

CA-L-Serv provides ADDFILE options that you can use to override VSAM defaults:

Performance Value	ADDFILE parameter	Default Value
Number of data buffers	BUFND(<i>nnnnn</i>)	5
Number of index buffers	BUFNI(<i>nnnn</i>)	5
Number of VSAM placeholders (which limits the number of concurrent VSAM requests for a file)	STRNO(<i>nnn</i>)	5

For example, to allocate 32 data buffers and 32 index buffers in the private buffer pool for file DPMFPDF2, specify this command in the parameter data set:

```
ADDFILE DPMFPDF2 BUFND(32) BUFNI(32)
```

Important! These options can cause I/O problems if used improperly. When the number of buffers and placeholders is too low, performance may suffer. When the number is too high, you may run out of virtual storage. Therefore, fully consider the impact on virtual storage and performance before specifying the BUFND, BUFNI, and STRNO options.

Performance Adjustments

You can improve I/O performance for most CA-L-Serv managed files by changing any of the following OPTION parameter values on the ADDFILE command:

What You Can Do	OPTION Value	Considerations
Defer write operations for a record until the VSAM buffer is needed for another operation or the deferred write interval expires.	DEFER	<ul style="list-style-type: none"> ■ Use DEFER only with LSR buffer pools. ■ Speeds updates, deletions, and insertions. ■ The deferred write interval is set through the MAXDORM parameter on the OPTIONS command. ■ The most recent version of a record may be lost if a system problem occurs before the data is physically written to a disk.
Pad short records with blanks until the records are <i>nn</i> bytes long.	MINLEN(<i>nn</i>)	<ul style="list-style-type: none"> ■ MINLEN is a safeguard to the TRIM option because it ensures that clients receive records in the length they expect. ■ Do not specify MINLEN unless directed to do so by the client documentation.
Trim trailing blanks from the data portion of variable length records.	TRIM	<ul style="list-style-type: none"> ■ Saves disk space by reducing the record length before the record is stored on disk. ■ Never affects the key portion of a record. ■ Do not use TRIM if the VSAM data set has been defined with fixed length records.

For example, to set performance options for the file DPMFSIF, specify the following command in the CA-L-Serv parameter data set:

```
ADDFILE DPMFSIF OPTION(TRIM,DEFER) POOL(3)
```

Important! Misuse of these options can cause serious I/O or other system problems. CA Technologies strongly recommends that you use the default values provided by the client application that uses the file.

File Groups

A file group is a set of files that contains data used by a client application. Client applications can use file groups to record, access, and permanently store various types of data. For example, CA Endeavor Software Change Manager uses file groups to keep journals and to perform point in time recovery based on journal data.

When a client uses file groups, CA-L-Serv manages those files on behalf of the client. CA-L-Serv determines which file in the group to use, records data in that file, and when the file becomes full, automatically switches to a new file.

CA-L-Serv can also archive the contents of all the files in a file group so that you have a permanent copy of the data.

Note these points about file groups:

- In a multiple system environment, file groups can contain data obtained from several systems. You do not need to make separate file groups available to each system.
- CA-L-Serv never overwrites data in one of these files. If all files in the group become full, CA-L-Serv issues a message and stops recording data in them.

Characteristics of Files in the Same File Group

When you define a file group, you assign it a group ID. All files associated with the same group ID are part of the same file group. There is no limit to the number of files per file group.

The files in a file group must have the following characteristics in common:

- File format (all KSDS or all ESDS)
- Maximum logical record length
- For KSDS files, key length and key offset.

Define File Groups

To define a file group, include the `GROUP` parameter on an `ADDFILE` command. For example, to place the `JRNL1` file under the management of CA-L-Serv and to associate it with file group `UGRPID`, specify this command in the CA-L-Serv parameter data set:

```
ADDFILE JRNL1 GROUP(UGRPID)
```

To add subsequent files to the group, issue an `ADDFILE` command specifying the same group ID for each file you are adding to the group.

These file management options must be identical for files in the same group:

- They must all use the same buffer pool. (That is, the value for the `POOL` parameter must be the same for all of those files.)
- If you specify `OPTION(DEFER)`, `OPTION(APPEND)`, or `OPTION(TRIM)` for one file in the group, specify that value for all files in the group.

Select Files and Groups

By default, the order in which you issue `ADDFILE` commands determines which file group and file CA-L-Serv examines first. CA-L-Serv selects the first empty file it finds and skips over files that already contain data. You can change the way CA-L-Serv examines and selects files:

- To examine a particular file group or file first, use the `SWITCHFILE` command (as described in the *Reference Guide*).
- To select a partially full file when it is the first available file, specify `OPTION(APPEND)` on the `ADDFILE` command. CA-L-Serv appends data after any existing data in the file.

Set File Access

The DISP parameter of the ADDFILE command determines whether the file server requires exclusive use of the file specified in the ADDFILE command. If you specify:

- SHR (default), the file server allows shared access to the file. The file may be allocated to other users, but VSAM sharing options apply. The VSAM share options must be set as follows, or integrity exposure exists: SHROPTION=(1,3) or SHROPTION=(2,3).
- OLD, the file server must have exclusive use of the file. The ADDFILE command fails if the file is already allocated to another job or user of the system. Once the file is allocated to CA-L-Serv, the SHAREOPTIONS specified when the VSAM data set was defined become irrelevant because no other job or user of the system will be able to access the data set.

Note: When no DISP is specified on the ADDFILE command, the disposition of the data sets defaults to DISP(SHR).

Example

For exclusive access to the DPMFPDF2 file, specify this command in the parameter data set:

```
ADDFILE DPMFPDF2 DISP(OLD)
```

Propagate ENQ Requests

To control access to files that you put under the management of CA-L-Serv, CA-L-Serv issues ENQ requests as follows:

ENQ Component	Value That CA-L-Serv Uses
Qname	LSERVDSN
Rname	Dsname of the managed file
Scope	SYSTEMS

In a multiple system environment, you must propagate these ENQ requests globally. If you are using a product that propagates ENQ requests globally (such as CA MIM), make sure it propagates ENQ requests for each of the files that you put under the management of CA-L-Serv to all systems that are part of the complex.

Protect Your Data Sets

When a client application issues a request to CA-L-Serv to OPEN a data set on its behalf, the external security package (CA Top Secret, CA ACF2 or CA-L-Serv: RACF) is invoked to verify that the user has the required level of authority. This is done by checking the access or the user against the file using the CA-L-Serv specific resource class of \$LSRVDSN.

Before CA-L-Serv is allowed to open a data set placed under its control by an ADDFILE command, external security is invoked to verify that CA-L-Serv has the required authority.

Manage the File Server

When system outages occur, CA-L-Serv lets you display information about the Host and Remote file servers and respond appropriately to the system outage.

Display Information about the File Server

To display information about the file server, issue a DISPLAY command. Specify one or more of these parameters when using the TASK(FILESERVER) parameter:

DISPLAY Parameter	What It Does
ALL	Displays all available information for the file server.
BUFFERPOOL	Displays information about LSR buffer pools for files that CA-L-Serv is managing. (See message LDM0552I.)
DATABASE	<p>Displays information about each file that CA-L-Serv is managing, including options set through the ADDFILE command and the file status. (See message LDM0522I). By default, the display includes all managed files. To change the display, issue the command as follows:</p> <ul style="list-style-type: none"> ■ To display only one file, specify its name on the DDNAME parameter. ■ To display only one file group, specify a group ID on the GROUP parameter. ■ To display cumulative information and then reset the display, specify RESET.

DISPLAY Parameter	What It Does
OPTIONS	Displays file server values that you can change while the server is running. These values include data buffer size, subsystem name for the communications server, maximum write buffer interval, and server type. (See message LDM0526I).
STATISTICS	<p>Displays statistics about I/O activity for VSAM files. (See message LDM0546I.) By default, CA-L-Serv displays a line for each file. Information is cumulative as of the last time the statistics display was reset. To change the display, issue the command as follows:</p> <ul style="list-style-type: none"> ■ To limit the display to a particular file, specify its name on the DDNAME parameter. ■ To display cumulative statistics and then reset the display, specify RESET. ■ To display an additional line that shows average service times, specify =SERVICE.
SYSTEMS	Displays a list of systems where the file server is running. For each system, the display includes statistics about send and receive requests, what type of server is running, and how many times CA-L-Serv split and reassembled data to fit its data buffers. (See message LDM0555I.)

For example, to display the operating values of the file server, issue the following command:

```
DISPLAY TASK(FILESERVER) OPTIONS
```

Respond to System Outages

When a CA-L-Serv system experiences an outage, the following occurs:

- When a remote system fails, clients on other systems can continue to access and transmit data through CA-L-Serv.
- When a host system fails, all systems forwarding I/O requests to the host are affected by the outage. Clients cannot access files that the host server is managing, no matter what system those clients run on.

What You Should Do

The following table tells you how to respond to a system outage. If you can tolerate a delay in accessing files on a failed host system, follow the procedure described under Short Outages. If you cannot tolerate a delay, follow the procedure described under Long Outages.

Type of System	Length of Outage	Procedure for Responding
Host	Short	After the host system becomes active again, restart CA-L-Serv on that system.
	Long	See procedure that follows.
Remote	Short or Long	After the remote system becomes active again, restart CA-L-Serv on that system. Make sure you issue any CA-L-Serv commands that you normally issue at startup time.

To respond to a host type of system with a long outage length

1. Choose a new host system.
2. Stop the file server on the new host system by issuing the following command:
DETACH FILESERVER
3. Restart the file server on the new host system by issuing the following command:
ATTACH FILESERVER SERVERTYPE(HOST)
4. On the new host system, issue CA-L-Serv startup commands that you would normally issue on a host system. (Including ADDPOOL and ADDFILE commands.)

Maintain Managed Files

CA-L-Serv gives you control over the maintenance of the managed files.

Set File Availability

- You can put a file on hold, causing CA-L-Serv to put requests for that file into a queue until the file is released. To do this, issue this command:

```
HOLDFILE ddname
```

When you are ready to restore normal I/O processing for the file, you can release it by issuing this command:

```
RELEASEFILE ddname
```

- You can make a file unavailable to any job, so that all requests for it fail, by issuing this command:

```
CLOSEFILE ddname
```

When you are ready to restore normal I/O processing for the file, you can make the file available by issuing this command:

```
OPENFILE ddname
```

- You can remove a CA-L-Serv file from the management of CA-L-Serv by issuing:

```
REMOVEFILE ddname
```

This command does not delete a file it simply removes the file from the management of CA-L-Serv. The file is first closed, and then deallocated (if it was dynamically allocated).

Important! The `HOLDFILE`, and `CLOSEFILE` commands are not needed under normal circumstances. Use them only in consultation with CA Technologies Support.

The LDMAMS Utility

You can use the LDMAMS utility to perform the following operations on each file managed by CA-L-Serv:

- Back up the file
- Restore the file from a backup copy
- Delete the contents of the file
- Compress the file
- Archive a file group or a file group member

JCL for the LDMAMS Batch Job

To run the LDMAMS utility as a batch job, use the following JCL:

```
//stepname EXEC PGM=LDMAMS
//SSN$ssname DD DUMMY
//SYSPRINT DD SYSOUT=cclass
//SYSIN DD dsn specification or * control statement(s)
//ddname DD dd parameters for sequential files
```

In this JCL, provide the following information:

- The subsystem name (ssname) of the CA-L-Serv started task with which you are communicating. See CA-L-Serv Configuration in this chapter for more information about identifying the CA-L-Serv subsystem name.
- One or more control statements in the SYSIN data set. The control statements execute the operations listed above. A statement may be specified either by pointing to a data set where the statements reside, or by listing an asterisk followed by the statement itself.
- DD statements to define the sequential data sets that provide or receive data, depending upon the LDMAMS operations you are performing.

Note: The LDMAMS JCL does not contain DD statements for the files managed by the file server. These files are identified in the SYSIN statements. Adding DD statements for these data sets will cause the LDMAMS job to fail.

Back Up and Restoring Files

You can use the REPRO statement of the LDMAMS utility to create backup copies and restore the contents of CA-L-Serve managed files.

Back Up Managed Files

Using the CA-L-Serv REPRO statement of the LDMAMS utility, you can backup a VSAM file managed by providing the following:

- The ddname of the source file, which is the VSAM file you are backing up. Specify this ddname on the INFILE parameter. The ddname must match the ddname on the ADDFILE command. Do not specify a DD statement for managed data sets in the LDMAMS JCL.
- The ddname of the target file, which is the sequential file where the backup copy is stored. Specify this ddname on the CA-L-Serv OUTFILE parameter. The LDMAMS JCL must include a DD statement for this file.

For example, to back up FILE1 and store the backup copy in BCKFILE1, specify the following REPRO statement:

```
//          EXEC  PGM=LDMAMS
//SSN$xxxx DD    DUMMY
//SYSPRINT DD    SYSOUT=A
//SYSIN     DD    *
      REPRO  INFILE(FILE1)  OUTFILE(BCKFILE1)
//BCKFILE1 DD  DSN=LSERV.BACKUP, DISP=(NEW,KEEP) . . .
```

Restore a Managed File from a Backup Copy

You can use the REPRO statement to restore a VSAM CA-L-Serv file that has been backed up. On the REPRO statement, you must provide the following:

- The ddname of the source file (that is, the sequential file containing the backup copy). Specify this ddname on the CA-L-Serv INFILE parameter. The LDMAMS JCL must include a DD statement for this file. Do not specify a DD statement for managed data sets in the LDMAMS JCL.
- The ddname of the target file, which is the VSAM file you are restoring. Specify this ddname on the CA-L-Serv OUTFILE parameter. The ddname must match the ddname on the ADDFILE command of the file.

For example, to restore FILE1 from the backup copy contained in BCKFILE1, specify the following REPRO statement:

```
//          EXEC  PGM=LDMAMS
//SSN$xxxx DD    DUMMY
//SYSPRINT DD    SYSOUT=A
//SYSIN     DD    *
      REPRO  INFILE(BCKFILE1)  OUTFILE(FILE1)
//BCKFILE1 DD  DSN=LSERV.BACKUP, DISP=OLD. . .
```

Change Default Values for REPRO

The default procedures that REPRO performs are as follows:

- When copying records from a source file, start with the first record in the file and copy through to the last record.
- When copying records into a target file, append the records to the existing records in the file.
- When copying records into a target file, do not replace duplicate records.

You can change these default values by using these parameters with REPRO:

Parameter	Causes REPRO To
FROMKEY(<i>key</i>)	Start with the specified record.
TOKEY(<i>key</i>)	End with the specified record.
REUSE	Delete the contents of the target file before copying records into it.
REPLACE	Use the latest copy of a duplicate record.

For more information about REPRO statements, see the *Reference Guide*.

Specify Records to Back Up

Specify the following REPRO statement to back up all records in a file between and including the key values of "MARCH" and "JUNE":

Note: In collating sequence, JUNE comes before MARCH, so the FROMKEY is set to "JUNE" and the TOKEY is set to "MARCH."

```
//          EXEC  PGM=LDMAMS
//SSN$xxxx DD   DUMMY
//SYSPRINT DD   SYSOUT=A
//SYSIN   DD    *
          REPRO INFILE(FILE1) OUTFILE(BCKFILE1) FROMKEY(JUNE) TOKEY(MARCH)
//BCKFILE1 DD   DSN=LSERV.BACKUP
```

Delete the Contents of Files

Use the RESET statement to delete the contents of managed data sets. Do this by providing the ddname of the file that should be emptied on the OUTFILE parameter of the RESET statement, which is described in the *Reference Guide*. This ddname must match the ddname on the ADDFILE command.

For example, to delete the contents of FILE1, specify a RESET statement as follows:

```
//          EXEC  PGM=LDMAMS
//SSN$xxxx DD    DUMMY
//SYSPRINT DD    SYSOUT=A
//SYSIN   DD     *
          RESET  OUTFILE(FILE1)
//*
```

Note: A file can be reset only if it was defined with the REUSE option of the DEFINE CLUSTER control statement of the IDCAMS utility.

Compress Files

When compressing a managed CA-L-Serv file, CA-L-Serv performs the equivalent of the backup, reset, and restore operations in one step. CA-L-Serv backs up the managed file to a sequential file, deletes the contents of the managed file, and then copies data from the sequential file to the managed file. The compression process:

- Backs up and restores the file without allowing it to be changed between operations. When CA-L-Serv compresses the file, it holds the file exclusively until the backup, reset, and restore operations have completed. Do not specify a DD statement for managed data sets in the LDMAMS JCL.
- Causes the data to be reorganized more efficiently. Compression eliminates VSAM control interval and control area splits; however, it does not automatically release secondary extents.

COMPRESS Statement

You can use the COMPRESS statement to perform a compress operation by providing the following information:

- The ddname of the file that is to be compressed (on the CA-L-Serv INFILE parameter). This must match the ddname on the ADDFILE command.
- The ddname of the sequential file that temporarily holds the backup copy (on the CA-L-Serv WORKFILE parameter). The LDMAMS JCL must include a DD statement for this file.

For example, to compress the file FILE1 using a work file called TEMP1, specify a COMPRESS statement as follows:

```
//          EXEC  PGM=LDMAMS
//SSN$xxxx DD    DUMMY
//SYSPRINT DD    SYSOUT=A
//SYSIN     DD    *
COMPRESS INFILE(FILE1)  WORKFILE(TEMP1)
//TEMP1    DD    DSN=LSERV.TEMP
```

Note: A file can be compressed only if it was defined with the REUSE option of the DEFINE CLUSTER control statement of the IDCAMS utility.

Archive Files and File Groups

You can permanently save the contents of a CA-L-Serv file group or one of its CA-L-Serv files by archiving the contents of the file. In an archive process, CA-L-Serv copies data from the file and saves the data in a sequential file. Then, CA-L-Serv empties the original file so that it can be used again.

You can archive a single file or all non empty files in a file group.

ARCHIVE Statement

To archive members of a file group, specify an ARCHIVE statement and provide the following information:

- The ddname of the file that you are archiving (using the INFILE parameter) or the ID of the file group you are archiving (using the CA-L-Serv GROUP parameter).
- The sequential file where the copied data is stored (using the CA-L-Serv OUTFILE parameter). The LDMAMS JCL must include a DD statement for this file.
- An indication as to whether the currently active file is to be archived (using the CA-L-Serv SWITCH parameter). By default, CA-L-Serv does not archive a file that is currently being used.

Note: A file can be archived only if it was defined with the REUSE option of the DEFINE CLUSTER control statement of the IDCAMS utility.

For example, to archive all files in the ENDJRN group (including the currently active file) and store their contents in the ARCHDS file, specify the following ARCHIVE statement:

```
//          EXEC  PGM=LDMAMS
//SSN$xxxx DD    DUMMY
//SYSPRINT DD   SYSOUT=A
//SYSIN      DD   *
ARCHIVE     GROUP(ENDJRN)  OUTFILE(ARCHDS)  SWITCH
//ARCHDS    DD    DSN=LSERV.ARCH
```

Or, to archive file JRNL1 and store its contents in the ARCHDS file, specify the following ARCHIVE statement:

```
//          EXEC  PGM=LDMAMS
//SSN$xxxx DD    DUMMY
//SYSPRINT DD   SYSOUT=A
//SYSIN      DD   *
ARCHIVE     INFILE(JRNL1)  OUTFILE(ARCHDS)
//ARCHDS    DD    DSN=LSERV.ARCH
```

Automatic Archive of File Groups

When a file is assigned to a file group, you can specify the SUBMIT option on the ADDFILE command. This option automatically causes a JCL to be submitted when the file is full. When using the SUBMIT option you must also use the JCLMEMBER keyword on the ADDFILE command to indicate the name of the JCL which will be submitted when the data set becomes full.

The JCL specified on the JCLMEMBER keyword must be a member of the concatenation specified in the JCLLIB DD statement in your CA-L-Serv procedure.

ADDFILE Command with the SUBMIT OPTION

Specify the following ADDFILE statements to cause files in the NDJG group to be archived when they become full:

```
ADDFILE J1 END.JRN1 GROUP(NDJG) OPTION(SUBMIT) JCLMEMBER(ENDARCH)
ADDFILE J2 END.JRN2 GROUP(NDJG) OPTION(SUBMIT) JCLMEMBER(ENDARCH)
ADDFILE J3 END.JRN3 GROUP(NDJG) OPTION(SUBMIT) JCLMEMBER(ENDARCH)
```

Note: As explained in the following section, it is possible to use the same JCL member to archive different files.

Generic JCL for Archiving Files

Using symbolic variables, you can create generic JCL that can be used to archive many different files. If you set up generic JCL using symbolic variables, you can use the same JCL for every managed data set, rather than having to create a specific archiving procedure for each file.

Symbolic variables will be replaced by CA-L-Serv before submitting the JCL. CA-L-Serv provides the following symbolic variables:

Substitution Value	Symbolic Variable
The ddname of a file you are archiving	%FILE%
The name of a file group you are archiving	%GROUP%
The ddname for the sequential file where you store the data you are archiving	%DSN%

Note: Do not use symbolic variables in JCL that you submit yourself. Use them only in JCL that CA-L-Serv submits automatically.

For instance, suppose you add the files F1 through F10 to the file group G100, specify SUBMIT with each, and use ARCHJCL to perform the archiving procedure. To place the files in a data set that follows the naming convention of LSERV.ARCH.group.file, create ARCHJCL using the following statements:

```
//          EXEC  PGM=LDMAMS
//SSN$xxxx DD   DUMMY
//SYSPRINT DD   SYSOUT=A
//SYSIN     DD   *
  ARCHIVE  INFILE(%FILE%)  OUTFILE(ARCHDS)
//ARCHDS   DD   DSN=LSERV.ARCH.%GROUP%.%FILE% ...
After F3 is archived, you can find it in LSERV.ARCH.G100.F3.
```

Note: For more information about using the SUBMIT option with the ADDFILE command, see the *Reference Guide*.

Use the Communications Server

The communications server controls the cross system communication service. When using cross system communication, client applications may do any of the following:

- Transmit client data across z/OS systems. For example, a client application may transmit commands and WTOs to copies of that same client application on other z/OS systems.
- Share client files across z/OS systems. For example, a client application may transmit journal data among systems. By doing this, data from several systems can be maintained in one set of journals.

Communication Protocols

The communications server supports three protocols: XCF, VTAM LU 6.2, and VTAM LU 0.

Cross System Coupling Facility (XCF)

The XCF protocol provides reliable, high speed communication between systems in a sysplex environment. A sysplex (systems complex) is a set of z/OS systems that have been given a sysplex name and authority to use XCF services. Since XCF provides the fastest and most reliable method for data transmission, CA strongly recommends that you use it where available.

Virtual Telecommunications Access Method (VTAM)

You can use either VTAM LU 6.2 or VTAM LU 0 for communication between z/OS systems that are not part of a sysplex. If you have some z/OS systems that are within a sysplex and others that rely on VTAM, you can implement both methods within the same communications server.

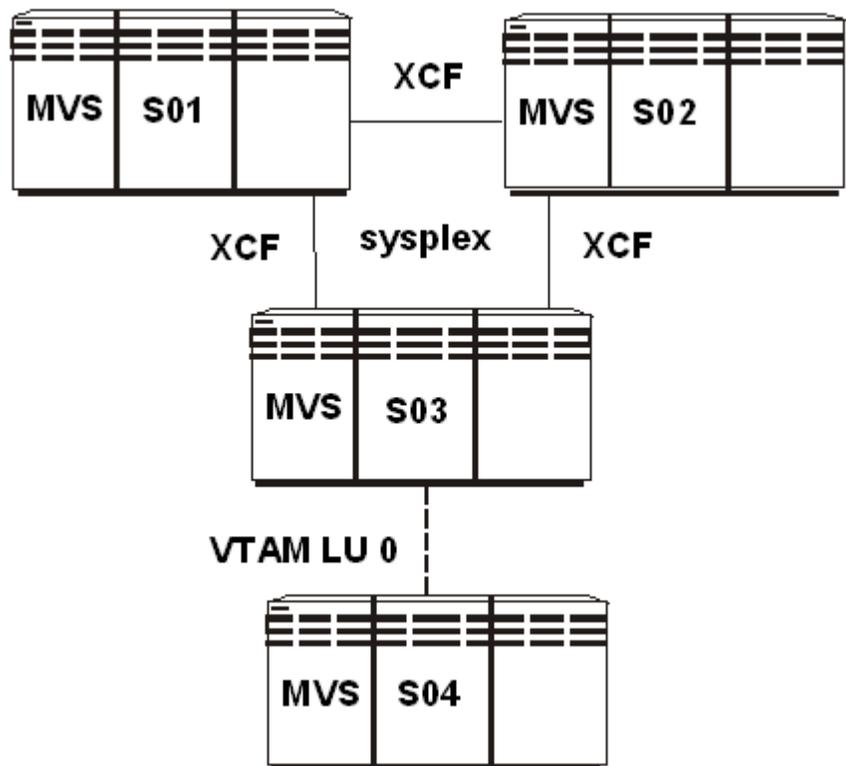
Important! The same z/OS system can simultaneously use both the XCF protocol and the VTAM protocol; however, communication between any two z/OS systems must be exclusively handled using either the XCF protocol or one of the VTAM protocols.

Use of Multiple Communication Services and Protocols

You can use multiple communication services and protocols on the same system. But you can use only one protocol - XCF, VTAM LU 6.2, or VTAM LU 0 - for communication between any two systems at any one time.

A Typical Implementation

The use of multiple communication services and protocols is illustrated below. The z/OS systems S01, S02, and S03 are in a sysplex, while z/OS system S04 is not. XCF is used for communication among S01, S02, and S03, while VTAM LU 0 is used for communication between S03 and S04.



The communication setup shown above allows the following communication between CA-L-Serv applications:

- A CA-L-Serv application running on system S01, S02, or S03 can communicate with any CA-L-Serv application running on system S01, S02, and S03.
- A CA-L-Serv application running on system S03 can communicate with a CA-L-Serv application running on system S04.

Note: A CA-L-Serv application running on S01 or S02 cannot communicate with a CA-L-Serv application running on S04. That is, S03 does not act as a gateway between S01/S02 and S04.

Set Up the Communications Server

There are several setup tasks associated with the communications server. The exact tasks that must be performed on each system depend on the communication services and protocols used on that system.

For each communication service/protocol combination in use on a system, see the appropriate section listed below:

Service/Protocol	Where Setup Tasks Are Described
XCF	See Configuring XCF Communication
VTAM	See Configuring VTAM Communication

Note: Do not start the communications server before performing all of the appropriate setup tasks described in the sections listed in the above table.

Start and Stop the Communications Server

To start the communications server, issue an ATTACH COMMSERVER command. This command can be issued with various options, depending upon the communication services and protocols that you want to use on that system. See the Reference Guide for more information on the ATTACH command.

To stop the communications server, issue a DETACH COMMSERVER command:

```
DETACH COMMSERVER
```

To issue the DETACH command from a command member, specify:

```
READ member
```

member

Contains the necessary DETACH command.

Display Information about the Communications Server

To display information about the communications server, issue a DISPLAY command. Specify one or more of these parameters:

APPLICATIONS

Displays a list of the client applications currently connected to the local communications server, including the number of send and receive requests from each client. (See message LDM0920I.)

TASK(COMMSERVER) INIT

Displays communications server values that cannot be changed while the server is running. These values include ACB name, maximum data size, and the size of transmission buffers. (See message LDM0911I.)

TASK(COMMSERVER) OPTIONS

Displays communications server values that you can change while the server is running. These values include communications service and protocol, transmission buffers per client, auxiliary message log name, retry specifications, and send requests per client. (See message LDM0910I.)

ROUTES

Displays information about communication routes. (See message LDM0912I.) By default, the display includes all routes.

To display only active routes, specify ROUTES(ACTIVE).

To display only inactive routes, specify ROUTES(INACTIVE).

For example, to display all of the operating values of the communications server, issue the following command:

```
DISPLAY TASK(COMMSERVER) INIT OPTIONS
```

Configure XCF Communication

Before performing any other configuration tasks, configure the sysplex for those systems that will use XCF. See your current IBM z/OS: *Initialization and Tuning Guide* and *z/OS Planning: Sysplex Management* documents for instructions on this task.

Start the Communications Server with XCF

If you plan to use only XCF communication, issue the ATTACH command as follows:

```
ATTACH COMMSERVER XCF(YES)
```

Start the Communications Server with XCF and VTAM

If you plan to use both XCF and VTAM, issue the ATTACH command as follows:

```
ATTACH COMMSERVER ACBNAME(name) CONTYPE(luvalue) XCF(YES)
```

name

Defines the applid that identifies CA-L-Serv to VTAM on this system.

luvalue

Either LU0 (for LU 0) or LU62 (for LU 6.2); LU0 is used by default.

Establish XCF Communication Routes

When using the XCF protocol, the communications server on each member of the sysplex automatically attempts to establish a communication route with every other member of the sysplex. You cannot manually activate or deactivate XCF communication routes, nor is there a need to manually activate or deactivate XCF communication routes.

The status of a communication route between any two systems is dependent upon the state of the communications server on each system:

- If a communications server has been started on only one member of a communicating pair, the communication route between the two members is visible (using the DISPLAY command) only on the member with the started communications server, and the route is inactive.
- When the communications server has been started on both members of a communicating pair, the communication route between the two members is visible on both systems, and the route is active.

Use the Same Subsystem Names

The subsystem name must be the same on all systems that will communicate using XCF. The subsystem name is set using the SSNM parameter in the CA-L-Serv startup procedure.

Configure Data Transmission Values

The communications server uses internal data buffers to temporarily store outgoing and incoming data, as well as delivery instructions for the data. These buffers are called *transmission buffers*.

Client application issues send requests to the communications server, which transmits data on their behalf. Data remains in buffers until a transaction is completed. The communications server transmits outgoing data immediately, and it relinquishes incoming data when the appropriate client issues a receive request for that data.

Configurable Elements

You can adjust the following data transmission values when using XCF communication:

- The maximum amount of data sent per transmission.
- The number of transmission buffers per client.
- The maximum number of outstanding send requests per client.

Note: The values set here apply to all CA-L-Serv communication (for both XCF and VTAM). Review the following sections for additional information about adjusting the default values for these settings.

Maximum Data Size During Transmission

If you know the maximum size of the data that clients are transmitting, you can minimize storage used for communication. To do this, specify the maximum data size (in kilobytes) on the MAXSENDSIZE parameter of the ATTACH command. If a client tries to transmit more than that amount of data, CA-L-Serv rejects the request.

For example, to transmit no more than 20K of data at a time, specify:

```
ATTACH COMMSERVER ACBNAME(NONE) MAXSENDSIZE(20) XCF(YES)
```

If you do not know the maximum size, use the default value 32.

Note: You cannot change the maximum data size while the communications server is running. You need to recycle the Communications Server using the DETACH and ATTACH commands for a new value to take effect.

The default value of 32K is suitable for most client applications. See the documentation for the application to determine if you need to adjust this value.

Other Data Transmission Values

There are additional data transmission values that can be adjusted. However, because the default values are probably suitable for your needs, do not change the defaults unless directed to do so by the documentation of your client application.

- **Transmission Buffers per Communications Server**

By default, CA-L-Serv allocates a maximum of ten buffers per client. Once this limit is reached, CA-L-Serv rejects additional incoming data for that client until the client receives the data waiting for it in these buffers. Do not change this default setting unless you experience storage problems.

If you do need to change this setting, specify a new value for the CA-L-Serv HOLDBUF parameter on the ATTACH command. (Through the OPTIONS command, you can change this value while the server is running.)

- **Outstanding Send Requests per Communications Server**

By default, CA-L-Serv allows each client ten outstanding send requests. Once this limit is reached, CA-L-Serv rejects additional outgoing data from that client until the data in the buffers has been transmitted. Do not change this default setting unless you experience storage problems.

If you do need to change this setting, specify a new value for the SENDLIMIT parameter on the ATTACH command. (Through the OPTIONS command, you can change this value while the server is running.)

Test Data Transmission

You can use the Communications Server Installation Verification procedure to verify that two systems are able to transmit data. See the *Installation Guide* for a full description of this procedure.

Respond to XCF Communication Problems

If you experience a communication problem when using XCF, issue the DISPLAY ROUTES command and see the appropriate situation below:

- No route appears between two systems.

Issue the following z/OS console command:

```
D XCF
```

Check that both systems are part of the sysplex. XCF can only be used between systems that are in the same sysplex.

- There is an inactive route between two systems.

Issue the following z/OS console command:

```
D XCF, GROUP, group
```

group

Defines the name associated with the XCF group used by CA-L-Serv.

The default group name is LSRVsubsystemname (LSRV prefixed to the subsystem name).

The output of this command lists all of the members of the specified group. By default, members are named LCOMsystemname (LCOM prefixed to the local system name). If a particular member does not appear in the output, it may indicate one of two problems:

- CA-L-Serv, the communications server is not running on the system associated with that member. Check that CA-L-Serv is running and that the communications server has been started with XCF enabled.
- The system was not assigned to the associated group. Ensure that the subsystem name is the same on that system as on the other systems in the group.

If these suggestions do not lead to a solution to your problem, contact CA Technologies Support.

Configure VTAM Communication

If you plan to use VTAM communication, you must provide information that enables CA-L-Serv to utilize VTAM. To do this, perform the tasks described in the *Installation Guide* before you start up the communications server.

Start the Communications Server with VTAM

If you plan to use a VTAM protocol only, issue the ATTACH command as follows:

```
ATTACH COMMSERVER ACBNAME(name) CONTYPE(luvalue)
```

name

Defines the applid that identifies CA-L-Serv to VTAM on this system. See the chapter "Starting CA-L-Serv" in the *Installation Guide* for instructions on setting the value of applid.

luvalue

Either LU0 (for LU 0) or LU62 (for LU 6.2). If CONTYPE is not specified, it defaults to LU0.

Start the Communications Server with XCF and VTAM

If you plan to use both XCF and VTAM, issue the ATTACH command as follows:

```
ATTACH COMMSERVER ACBNAME(name) CONTYPE(luvalue) XCF(YES)
```

name

Defines the applid that identifies CA-L-Serv to VTAM on this system.

luvalue

Either LU0 (for LU 0) or LU62 (for LU 6.2); LU0 is used by default.

Define VTAM Communication Routes

To transmit data between two systems using VTAM communication, each system must be defined as a node on the other system. When two systems have defined each other as nodes, a communication route can be opened between those systems.

Set Up a Route

To set up a communication route between system SYS01 (ACB name is P01) and system SYS02 (ACB name is P02), SYS01 must be defined as a node on SYS02, and SYS02 must be defined as node on SYS01.

To do this, issue the following ACTIVATE command on SYS01:

```
ACTIVATE P02
```

And issue the following ACTIVATE command on SYS02:

```
ACTIVATE P01
```

Override LU Version and Retry Specification

When you define a communication route, you can override the following values just for that route by adjusting one or both of the node definitions that make up the route:

- LU Type

The default value is set through the CONTYPE parameter of the ATTACH command on the communications server. You can override this default value by using the CONTYPE parameter on the ACTIVATE command. The LU type must be reset on both of the nodes that make up a VTAM communication route.

- Retry specifications for routes that become inactive

By default, the communications server does not try to reactivate inactive routes. If you want to automatically reactivate routes, you need to set the maximum number of retries and the retry interval. Default values are set through the RETRMAX and RETRY parameters on the ATTACH command. You can override these default values on the ACTIVATE command.

When you specify any of these parameters on an ATTACH command, they affect all VTAM routes originating from the corresponding system. When you specify them on an ACTIVATE command, they affect only the route associated with the specified node.

Deactivate a Communication Route

If you deactivate a VTAM path that the communications server is using, you need to deactivate the VTAM communication route associated with that path by deactivating one of the nodes that define the route. Use the DEACTIVATE command to deactivate a node. On the DEACTIVATE command, specify the name of the node corresponding to the route that you want to deactivate.

For example, to deactivate the route defined in the example shown previously, issue this DEACTIVATE command on SYS01:

```
DEACTIVATE P02
```

Or issue this DEACTIVATE command on SYS02:

```
DEACTIVATE P01
```

Note: When you use the DEACTIVATE command, the RETRY and RETRMAX values for the route are set to zero; this prevents attempts to reactivate a route that has been specifically deactivated. When you reactivate the route using the ACTIVATE command, if you do not manually set these values, they default to the values defined on the ATTACH command.

Configure Data Transmission Values

The communications server uses internal data buffers to temporarily store outgoing and incoming data, as well as delivery instructions for the data. These buffers are called *transmission buffers*.

Client application issues send requests to the communications server, which transmits data on their behalf. Data remains in buffers until a transaction is completed. The communications server transmits outgoing data immediately, and it relinquishes incoming data when the appropriate client issues a receive request for that data.

Configurable Elements

You can adjust the following data transmission values when using XCF communication:

- The maximum amount of data sent per transmission.
- The number of transmission buffers per client.
- The maximum number of outstanding send requests per client.

Note: The values set here apply to all CA-L-Serv communication (for both XCF and VTAM). Review the following sections for additional information about adjusting the default values for these settings.

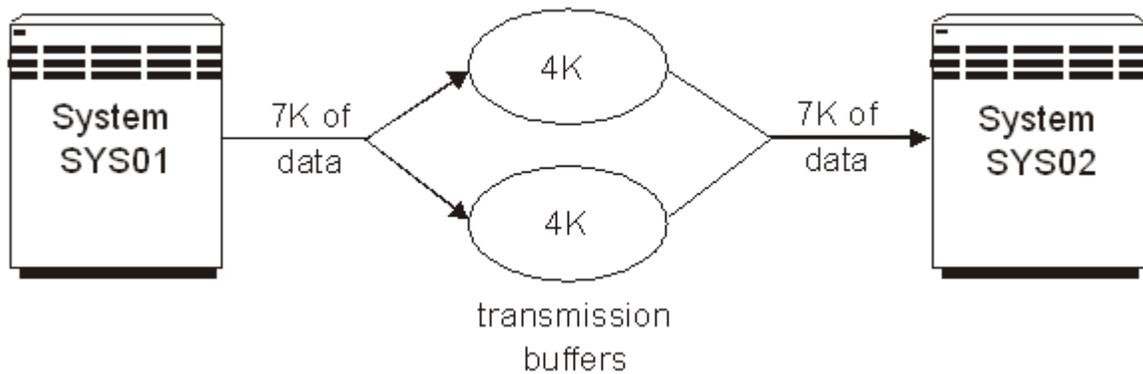
Effect of Buffer Size on Data Transmission

The performance of VTAM communication can be significantly affected by the size of the transmission buffers. As you set up the communications server, you must decide what buffer size is best by considering the following:

- Which set of clients are transmitting data through the server?
- What size data are they transmitting?

The data size is determined by the client and not by CA-L-Serv. If incoming data is too large to fit in a single transmission buffer, CA-L-Serv breaks it up and stores it in several buffers. CA-L-Serv recombines the data before delivering it to the receiving client.

For example, if a client is transmitting 7K of data and the transmission buffers are 4K each, CA-L-Serv breaks up and recombines data as shown here:



Since there is overhead involved in breaking up and recombining data, try to minimize the number of times CA-L-Serv does this by establishing the appropriate buffer size.

Determine the Best Buffer Size

Make your transmission buffers large enough to hold an average piece of transmitted data (not the largest piece of transmitted data). Do not make buffers too large, as this wastes storage space. Use the same buffer size on all systems.

By default, CA-L-Serv uses 4K buffers. To determine what buffer size you need, look at the types of data that CA-L-Serv is transmitting and the number of clients transmitting data through the same communications server:

- If clients are transmitting small pieces of data, 1K or 2K buffers may be appropriate.
- If clients are transmitting large pieces of data (for example, records from a file), you need larger buffers. For files, look at the LRECL values for this data on a file by file basis to estimate the best buffer size.
- If several clients are using the same communications server, weigh the needs of all of those clients. You cannot set different buffer sizes for different clients, so estimate the best overall buffer size.

See the client documentation for additional recommendations on the optimal buffer size.

Note: If you are using LU 0, the buffer size must not exceed the maximum transmit size allowed by the VTAM path or all sends and receives will fail.

Change the Buffer Size

To change the size of the transmission buffers through the RECBUFFSIZE parameter on the ATTACH command, specify one of these values:

Average Data Size	Value to Use
Less than 944 bytes	RECBUFFSIZE(1K)
Between 944 and 1,968 bytes	RECBUFFSIZE(2K)
Between 1,968 and 8,112 bytes	RECBUFFSIZE(4K)
Between 8,112 and 16,304 bytes	RECBUFFSIZE(8K)
Over 16,304 bytes	RECBUFFSIZE(16K)

Note: You cannot change the buffer size while the server is running.

Test Data Transmission

You can use the LDMAMS utility to test the VTAM communication routes you have created using the ACTIVATE command. Through this utility, you create a pair of test applications (one on each communicating system) and you transmit test data between them. Before you start, make sure that:

- The communications server and the file server are active on both systems.
- You have defined and activated the route you are testing.

For more information about starting CA-L-Serv, see the *Installation Guide*.

Respond to VTAM Communication Problems

If you experience a communication problem when using VTAM with the communications server, follow these guidelines:

- If VTAM fails on a system, restart VTAM as soon as possible. As soon as the VTAM resources are activated CA-L-Serv reestablishes communications.
- If a VTAM route fails but VTAM remains active, reactivate that route using the ACTIVATE command.
- Issue D NET,ID= commands on both systems to verify that your ACB VTAM definitions (ACB and CDRSC) are active.

Chapter 9: CA Global SubSystem

Note: The topics in this section assume that you have knowledge of REXX programming.

This section contains the following topics:

- [Overview](#) (see page 276)
- [Components](#) (see page 277)
- [Run CA-GSS](#) (see page 277)
- [Stop CA-GSS](#) (see page 278)
- [Initialization Parameters](#) (see page 278)
- [IMODs](#) (see page 282)
- [System Security](#) (see page 289)
- [How IMODs Execute](#) (see page 291)
- [IMOD Facilities](#) (see page 292)
- [Data Set Security](#) (see page 292)
- [IMOD Naming Conventions](#) (see page 292)
- [IMOD Editor](#) (see page 293)
- [REXX Language in CA-GSS](#) (see page 296)
- [User-defined Functions and ADDRESS Environments](#) (see page 315)
- [How IMODs Are Created](#) (see page 320)
- [Allocate New ISET Data Sets](#) (see page 321)
- [Define an ISET to CA-GSS](#) (see page 321)
- [Write an IMOD](#) (see page 322)
- [Compile an IMOD](#) (see page 323)
- [Load a Compiled IMOD](#) (see page 323)
- [Test a Loaded IMOD](#) (see page 324)
- [IMOD Execution Using SRVBATCH](#) (see page 326)
- [Example: IMOD for Processing Operator Commands](#) (see page 328)
- [Package IMODs as a Load Module](#) (see page 330)
- [Back Up IMODs](#) (see page 332)
- [Restore an ISET](#) (see page 332)
- [Restore a Single IMOD](#) (see page 332)
- [Migrate ISETs from Previous Releases](#) (see page 333)
- [Batch Maintenance of ISETs and ILOGs](#) (see page 333)

Overview

The CA Global SubSystem, CA-GSS, lets CA Technologies products communicate with one another using cross memory communications and access authorized operating system services. CA-GSS gives you quick access to information from different areas to determine how to use your system resources more efficiently.

Many CA Technologies products use CA-GSS services during their normal operation. For example, CA-GSS lets CA Jobtrac Job Management access data gathered by CA SYSVIEW Performance Management, and then use this information to determine how to schedule a particular job.

CA-GSS services provide connectivity by using a collection of one or more REXX subroutines that are edited, compiled, and executed as a single program. This collection of subroutines is called an IMOD.

IMODs can monitor, automate, and regulate system activities. CA Technologies products use IMODs to communicate among themselves, and to provide additional product functionality. You can use IMODs for, but not limited to, the following types of tasks:

- Automating system monitoring
- Regulating resources
- Reporting on system activities
- Developing batch reports

Components

CA-GSS provides a simplified communication interface that lets various CA Technologies products communicate easily, seamlessly, and reliably. The following components make such communication possible:

- GOAL (the CA-GSS anchor)—Provides a single identification point for all other CA-GSS components, and all CA Technologies products that use the CA-GSS services.
- CA-GSS Base Product—Provides authorized services for CA Technologies products to reduce the dependence on CA Technologies-supplied SVC routines. This includes communication between address spaces on a single z/OS system.

It comprises a set of service routines and data areas in the extended common services area (ECSA), and an active address space owning a system linkage index (LX).

- IMODs—Are compiled REXX-based routines executed by ISERVE. CA Technologies supplies many IMODs to perform numerous integral services. You can also create IMODs to interface with CA Technologies products or to provide locally maintained functions.
- ISERVE (IMOD services routine)—Enables execution of IMODs. A primary ISERVE runs in the CA-GSS address space. You can use secondary ISERVEs for isolation, chargeback, or testing purposes. ISERVE is also available as a single-user utility program (SRVBATCH) for the batch execution of IMODs.
- IMOD Editor—Is an ISPF-based editor program that enables easy creation and maintenance of IMODs. Included in the editor is an IMOD test facility.
- GoalNet—Links multiple ISERVEs on multiple systems into a single integrated environment.

Run CA-GSS

To run CA-GSS

Issue the following command:

```
START GSSA
```

Initialization should proceed rapidly; when it is complete, the following message appears:

```
SRV220 Version 02.08.mm: Initialization Complete (ssid)
```

Note: For more information about messages, see the *CA Common Services Message Reference Guide*.

Stop CA-GSS

To stop CA-GSS

Issue *one* of the following commands:

```
STOP GSSA
```

```
P GSSA
```

```
F GSSA, STOP
```

If CA-GSS does not stop within a few seconds, issue the following command:

```
F GSSA, STOP FORCE
```

If CA-GSS still does not stop, cancel the address space and inspect the JESLOG and ISRVLOG listings for diagnostic messages.

Initialization Parameters

The CA-GSS started task locates the member that contains the initialization parameters through &MEMBER and the PARMLIB DD statement. CAW0OPTN(RUNPARM) is the default data set member that contains the parameters. During CA-GSS initialization, ISERVE reads this member (and possibly others) in order to obtain information about CA-GSS operation.

Depending on your CA Technologies products, you may need to customize certain CA-GSS initialization parameters. Unless otherwise noted, you should specify parameters on an as-needed basis. The following table provides an overview of these parameters and helps you determine which ones you need to customize.

Note: For more information about CA-GSS initialization parameters, see the *Reference Guide*.

Parameter	Description	Required?
ADDRESS	Identifies an ADDRESS environment and its associated load module.	Required for any external environment that is being accessed in an IMOD using an ADDRESS instruction. Most products that CA-GSS supports require an ADDRESS statement.
ALTNAME	Defines an alternative name for an address.	Most products that CA-GSS supports (such as CA SYSVIEW Performance Management) may require an ALTNAME statement if you change the default name for an ADDRESS environment that the product supplies.
CMD	Adds a CA-GSS/ISERVE operator command.	Not required.
COMMAND	Defines a word that can be issued as an operator console command to execute an IMOD.	Recommended once for each CA-GSS/ISERVE operator command.
DB2PLAN	Identifies the DB2 plan that should be used for dynamic SQL requests.	Required if you are using the DB2() built-in function.
DUMP	Routes CA-GSS and REXX dumps to a specific user or SYSOUT class.	Required.
EDITOR	Defines default data sets for the IMOD editor and the CA-GSS/ISERVE Operator Control Panel.	Required.
EVENTS	Determines the size of the internal EVENTS table.	In most cases, the default value of 1,000 is sufficient. However, you should increase this value if you receive S502 abends during normal processing.
FUNCTION	Identifies an external function and its associated load module.	Required if you are running IBM Tivoli NetView or you have written your own functions.
GLOBVAL	Defines a REXX global variable and assigns it an initial value.	Not required.
GOALNET	Defines a GoalNet node.	Required if you are using the CA-GSS GoalNet component.
GOALNETLOCAL	Declares the GoalNet name for the local system.	Required if you are using the CA-GSS GoalNet component.

Parameter	Description	Required?
HELPDSN	Identifies the ISPF panel library that contains the help panels for CA-GSS.	Not required.
ILOG	Defines an ILOG data set.	Required if you are using ILOG data sets or CA Insight Database Performance Monitor for DB2 for z/OS.
INCLUDE	Nests parmlib members.	Not required.
INSIGHT	Provides parameters for the CA Insight Database Performance Monitor for DB2 for z/OS initialization IMOD.	If you are using CA Insight Database Performance Monitor for DB2 for z/OS, check your documentation to see whether your release of CA Insight Database Performance Monitor for DB2 for z/OS provides an initialization IMOD and, if so, what parameters to pass it.
ISET	Defines an ISET for loading IMODs.	Required by CA Jobtrac Job Management.
JESNODE	Specifies the name of the JES node where this CA-GSS started task is executing.	Required if you are using CA Jobtrac Job Management in a multisystem environment.
JOBTRAC	Provides parameters for the CA Jobtrac Job Management initialization IMOD.	If you are using CA Jobtrac Job Management, check your documentation to see whether your release of CA Jobtrac Job Management provides an initialization IMOD and, if so, what parameters to pass it.
LOGLINES	Determines the maximum number of lines written to the ISRVLOG.	If omitted, the maximum is determined by the system limit on spool data sets.
LOGON	Defines and activates the Logon Facility.	Not required.
MAXSTACK	Limits the default size of IMOD stacks.	Not required.
MAXXREQ	Determines the maximum number of active, cross memory requests.	Not required.
PRIMARY	Indicates whether this CA-GSS address space is the primary one.	Not required.
PRODUCT	Identifies installed CA Technologies products that require special CA-GSS support.	Required if you are using CA Insight Database Performance Monitor for DB2 for z/OS.
SCHEDULE	Executes a list of IMODs immediately after the system initializes.	Not required.

Parameter	Description	Required?
SECURITY	Indicates what type of security software is installed on your system.	Required if you are running CA-GSS on a system where SAF-compatible security is in use.
SLICE	Determines the maximum number of REXX clauses that an IMOD can execute before relinquishing control to another IMOD.	In most cases, the default value is sufficient. Default: 10000
SSID	Provides the subsystem name for another product with which CA-GSS interacts.	Required if you are using CA OPS/MVS Event Management and Automation or CA SYSVIEW Performance Management, or if the DB2 address space that CA-GSS communicates with has a subsystem name other than DSN.
SSNAME	Defines a subsystem name for this CA-GSS address space.	Required. The default value is ISRV. Do not change the value unless you are starting a secondary ISERVE.
STORAGE	Forces CA-GSS work areas into 24-bit addressable storage.	Required on MVS/XA systems.
TCP/IP	Enables use of the TCP/IP communications protocol.	Not required.
TRACE	Sends trace output to the CA-GSS ISRVLOG data set.	Not required.
USER	Provides parameters for the installation-provided initialization IMOD.	Not required.
VIEW	Provides parameters for the CA View initialization IMOD.	If you are using CA View, check your documentation to see whether your release of CA View provides an initialization IMOD and, if so, what parameters to pass it.
VIUNIT	Defines the unit name for virtual I/O (VIO) requests.	Required if you are using any unit name other than VIO in your z/OS system generation.
VTAMRESTART	Indicates what action CA-GSS should take if VTAM fails and restarts.	Required if you are using the CA-GSS GoalNet component.
WTO	Provides criteria for intercepting WTO messages and executing IMODs based on them.	Required if you are using either of these CA Technologies products: <ul style="list-style-type: none"> ■ CA Insight Database Performance Monitor for DB2 for z/OS ■ CA Jobtrac Job Management

IMODs

CA-GSS provides the following types of IMODs:

- Service Routines—IMODs that can be called to provide both services and information to client IMODs.
- Server IMODs—Commonly used services provided as subroutines.
- Special Purpose IMODs—IMODs that are executed under certain circumstances.

You can write IMODs using the REXX language to provide your own functions and ADDRESS environments. These IMODs can either interact with CA Technologies products or provide stand-alone functions.

IMODs are stored in ISETs. Each ISET is a VSAM KSDS data set that contains source code, control information, and object code.

Service Routines

The CA-GSS base product contains the following service routines that can be called to provide services and information to client IMODs:

Note: For more information about CA-GSS service routines, see the *Reference Guide*.

\$\$SRV_COMMAND

Executes a CA-GSS/ISERVE operator command.

\$\$SRV_DSTAB_STAT

Returns information about ISET names and associated data sets.

\$\$SRV_ENQ_INFO

Returns information about the IMOD enqueue facility.

\$\$SRV_IDR_LIST

Lists load module identification records (IDR) data.

\$\$SRV_ILOG_STAT

Returns the status of one or more ILOG files.

\$\$SRV_IMOD_INFO

Returns information about a selected IMOD.

\$\$SRV_IMOD_LIST

Returns a list of all IMODs.

\$\$SRV_JOB_INFO

Locates a job within on the operating system.

\$\$SRV_MVS_DATA

Returns information about the currently executing z/OS.

\$\$SRV_PDS_LIST

Returns a PDS directory listing.

\$\$SRV_READ_IMOD

Returns the source code for an IMOD.

\$\$SRV_SYS_STAT

Returns ISERVE status data.

\$\$SRV_SYS_STATX

Returns the same data as \$\$SRV_SYS_STAT, but in a form that can be more readily processed by REXX programs. The returned data is not printable.

\$\$SRV_TPCF

Returns information about tape drives using the Tape Preferencing and Control Facility (TPCF) of the CA MIA Tape Sharing product.

Returned Data

Frequently, a service routine returns data on the stack. Each stack record generally has a number of data fields separated by delimiters. To permit the largest possible variation in data content, the convention is to use the first character of each stack record as the delimiter for parsing the remainder of the record.

Example: Data Parsing

The following REXX instruction pulls a record from the stack and loads the first character in the variable, *d*. This variable is then used to parse the remainder of the record, extracting *w1*, *w2*, and *w3*. The dummy variable (.) is specified to allow for the future addition of more fields.

```
parse pull 1 d +1 w1 (d) w2 (d) w3 (d) .
```

Returned String

In addition to information returned on the stack, each service routine can return a string of information.

- If you call the routine as a function, the string is returned as the value of the function.
- If you call the routine as a subroutine (using the REXX CALL instruction), the string is returned in the RESULT special variable.

Server IMODs

A server IMOD is a commonly used service provided as a subroutine. It is an IMOD task that is started, does not terminate, and processes requests from other IMOD tasks. It permits the same code to be used by a variety of callers and ensures that the appropriate resources are always used in a compatible manner.

You can use server IMODs to manage and synchronize the use of certain data. You can use them to provide an initialized environment for high-performance execution of requests, or to serialize the use of a particular process or resource. Other possible uses for server IMODs are the following:

- To log data from a TSO/E REXX-based EXEC

You use the SRVCALL() function to invoke an IMOD that passes data to a server. The server owns a sequential data set where the data is logged. Individual TSO users do not need to allocate and synchronize the use of the data set. No TSO user needs authorization to access the data set. GoalNet provides automatic access for multiple CPU configurations.

- To update a VSAM data set from multiple address spaces

Rather than having to synchronize all access to the data set, you can give ownership to a server IMOD. The server IMOD honors requests for reading from and writing to the data set. Access to the server can be provided by the SRVIMOD subroutine, callable from any installation-provided program.

- To provide a REXX routine that is to execute as part of certain production jobs

This routine, an IMOD running under SRVBATCH, uses CA SYSVIEW Performance Management to read and analyze SYSOUT data queued by previous job steps. Information from this analysis is then passed to a server (using the SRVCALL() function) and is retained in a VSAM KSDS. Later, an IMOD (initiated from CA Jobtrac Job Management) can request the server to return specific data that is used to control the scheduling of additional jobs.

You can also provide additional IMODs to extract summary information for display using TSO and the operator console, and to produce printed reports.

Note: For information about CA-GSS functions and programs, see the *Reference Guide*.

Server IMOD Conventions

CA-GSS provides a set of conventions to assist you in maintaining an environment that supports server IMODs. These conventions are not intended to limit the implementation of server IMODs, but rather to minimize errors during implementation.

You can insulate yourself from many of these conventions and most of the coding by using the `$$SERVER()` function. This function is an IMOD called as a subroutine to initialize, terminate, and communicate with server IMODs. The `$$SERVER()` function provides a straightforward means of requesting service from a server IMOD.

Note: For more information about the `$$SERVER()` function, see the *Reference Guide*.

Command Support

Server IMODs that follow the established conventions are supported by the `SERVER CA-GSS/ISERVE` operator command.

Global Variables

Global stem variables used to maintain the server IMOD environment are prefixed `&$$SERV_` and have the service name as the stem index (shown as *service* in the following list). The following global variable names are used to maintain the server IMOD environment:

&\$\$SERV_IMOD.service

Specifies the name of the IMOD that is invoked as the server.

&\$\$SERV_ID.service

Specifies the IMOD ID of the running server task. When the server is not running, the value of this variable is null.

&\$\$SERV_DESC.service

Describes the service being provided.

&\$\$SERV_FLAGS.service

Contains flags that control the operation of the server. Flags are one character (0 or 1) and positional, and indicate the following:

Flag	Description
1...	Server is accepting requests.
.1..	Server is processing requests.
..1.	Commands are being processed in AUTO mode.
...1	Server is shutting down.

&\$SERV_REQ.service

Contains the request string currently being processed.

Enqueues

To avoid conflicts that would result if multiple servers were to exist for the same service, an enqueue is obtained before starting a server, using the name of the service provider for *qname* and the name of the service for *rname*.

Stack Usage

During initialization, a server IMOD must declare (using the PUBSTACK() function) that stacks 1 and 2 are public, at least to the extent of granting QUEUE access.

The server IMOD must monitor both stacks 1 and 2. Records queued to Stack 1 are requests for service. Records queued to Stack 2 are commands that affect the functioning of the server.

Request Records

Request records are queued to Stack 1 to define the request. They should be transferred as a group (using the QUEUES() function) to prevent processing a partial request. Requests span multiple records and have the following format:

- Record 1 defines the request and the requester. All fields are required and separated by blanks. When a field is omitted, it must contain an asterisk (*) as a placeholder. The record has the following fields:
 - REQUEST
 - IMOD ID of the requester
 - Public stack of the requester to receive the results or * if no acknowledgment is required
 - Number of stack data records being passed
 - Request text
- Records 2 through *n* contain stack data to be processed by the server IMOD.

Command Records

Command records are queued to Stack 2 to define the commands. Each record contains a complete command. Each command record has the following fields, separated by blanks:

- COMMAND
- IMOD ID of the requester
- Public stack of the requester to receive the results or * if no acknowledgment is required
- Command to be executed
- Additional information as required by the command

Command processing takes preference over request processing.

Commands

The following commands should be supported by all server IMODs. These services are available through \$SERVER(INTERNAL,...) processing and supported in AUTO mode.

DEQUEUE

Transfers the contents of the request stack to the IMOD task issuing the command.

HOLD

Permits requests to be queued but does not process any requests from the queue.

PAUSE

Stops accepting further requests and defers the processing of any requests already queued.

PURGE

Fails each request in the current request stack and notifies the requester.

QUIESCE

Stops accepting further requests but continues to process any requests already queued.

RESET

Deletes all entries from the request stack. No processing is performed and the requesters are not notified.

RESUME

Cancels the effects of any outstanding HOLD, PAUSE, or QUIESCE.

STOP

Performs the QUIESCE and TERMINATE operations when all queued requests have been completed.

TERMINATE

Terminates processing immediately. Ignore any unprocessed requests.

Returning Results

The server IMOD should accumulate all results on a local stack. Then it should transfer the entire stack contents (using the QUEUES() function) to the requester, specifying POST.

The first record should be considered equivalent to RESULT and is the result returned if access is through the \$SERVER() function. Remaining records, if any, are returned to the requester using the stack.

Special Purpose IMODs

CA-GSS provides the following special purpose IMODs that are executed under certain circumstances. For example, the INITIALIZATION IMOD is executed once at system initialization. If you want to perform your own processing at this time, you must provide an IMOD of that name.

Note: For more information about special purpose IMODs, see the *Reference Guide*.

\$USER_ILOG_FULL

Is executed each time an ILOG data set is full.

INITIALIZATION

Is executed at ISERVE initialization.

TERMINATION

Is executed at ISERVE termination.

System Security

IMODs executing in a CA-GSS address space can access and update a variety of data sets and data areas. To prevent unauthorized activity, CA-GSS supports system security software that is compatible with the System Authorization Facility (SAF), including CA ACF2, CA Top Secret, and IBM RACF.

In z/OS, each task operates under the control of an access control environment element (ACEE), which controls access to all resources. SAF-compatible security software maintains the ACEE based on a user ID and ensures that the necessary checks are provided. CA-GSS ensures that an appropriate ACEE is in place for each executing IMOD and that all services invoked on behalf of the IMOD execute under the scope of that ACEE.

User IDs

CA-GSS needs two user IDs defined to your security software for proper security enforcement:

- The primary user ID assigned by the system to the CA-GSS started task or job
- A user ID with limited scope to use as a default ID for service requests that either have no associated user ID or for which CA-GSS cannot determine the associated user ID

You define a user ID as the default using the SECURITY initialization parameter.

Note: CA Technologies strongly recommends that you assign a specific user ID for CA-GSS.

- Because this is the default user ID, the ID should be very limited in scope.

How CA-GSS Uses User IDs

CA-GSS executes under its own primary user ID during initialization and when performing some housekeeping functions.

When an IMOD task is created, CA-GSS assigns it a valid user ID and obtains an ACEE for it. Generally, the user ID is taken from the task that triggered the IMOD task. For example, a request from a TSO user (IMOD editor or SRVCALL() function) is assigned the TSO user ID.

Sometimes, CA-GSS cannot determine what user ID to use. For example, if a started task, which does not have a user ID, issues a WTO that triggers an IMOD. In these cases, CA-GSS assigns the default user ID to the IMOD task.

Note: If no default user ID was defined at CA-GSS initialization, the IMOD task executes under the scope of the primary user ID of the CA-GSS address space.

Note: CA Technologies strongly suggests that you assign a specific user ID for CA-GSS.

The following table shows how CA-GSS determines the user ID under which an IMOD executes.

Type of IMOD	How CA-GSS Determines User ID
IMODs that support operator commands	If you have defined a user ID through the COMMAND initialization parameter in the CAW0OPTN(RUNPARM) data set member, CA-GSS uses it. Otherwise, CA-GSS uses its default user ID.
WTO-triggered IMODs	If you have defined a user ID through the WTO parameter in the CAW0OPTN(RUNPARM) data set member, CA-GSS uses it. If not, CA-GSS tries to determine and use the user ID of the WTO issuer. As a last resort, CA-GSS uses its default user ID. Note: During execution, an IMOD task can switch user ID by supplying a new user ID and its associated password through the SECURITY() function.
WTO-triggered IMODs with address space identifiers (ASIDs) matching those on the MONITOR CA-GSS/ISERVE operator command	CA-GSS tries to determine and use the user ID of the WTO issuer. Otherwise, CA-GSS uses its default user ID.
IMODs supporting the Logon Facility	If a user ID and password are provided, CA-GSS uses them. Otherwise, CA-GSS uses its default user ID.
Server IMOD	CA-GSS uses the user ID of the IMOD that started it.
ADDRESS environments and subtask	CA-GSS uses the user ID of the IMOD that invokes them. Note: If a subtask is reassigned to another IMOD, the user ID changes.

How IMODs Execute

IMODs are executed by IISERVE under the control of CA-GSS or as batch programs using SRVBATCH.

CA-GSS initiates an IISERVE in its address space. This is the primary IISERVE. You can also have secondary IISERVEs in other address spaces. The primary IISERVE processes all requests not specifically routed to secondary IISERVEs.

Note: Secondary IISERVEs are useful for testing or chargeback purposes.

IMODs are executed in the following ways:

- At the direction of certain CA Technologies products:
 - CA Deliver
 - CA Insight Database Performance Monitor for DB2 for z/OS
 - CA Jobtrac Job Management
 - CA OPS/MVS Event Management and Automation
 - CA SYSVIEW Performance Management
 - CA View
- At the request of CA agents running on non-z/OS platforms (such as AS/400, HP/9000, RS/6000, Windows, and others)
- At the request of another IMOD using an external subroutine call
- In response to installation-defined operator console commands
- In response to WTO messages that serve as triggers
- At the request of a TSO user from the IMOD editor test facility
- As batch programs using the SRVBATCH program
- At the request of TSO/E REXX (or compatible) EXECs using the SRVCALL() function
- To support a user session under the Logon Facility

IMOD Facilities

The following IMOD facilities enable you to create, modify, and execute IMODs:

- An ISPF-based editor called SRVEDIT
- A compiler called SRVCOMP
- A batch maintenance program called SRVMAINT
- Execution facilities called ISERVE

Note: Before writing an IMOD, you must allocate an ISET data set to contain it.

Data Set Security

When you invoke the IMOD editor, it builds a list of available ISETs using the CAW0OPTN(ISETS) member. An asterisk (*) in front of the description indicates that the ISET is read only. You can override the access type when you select an ISET.

Many installations secure data sets with RACF or other security software. When you try to update an IMOD, the IMOD editor tries to access an ISET in VSAM update mode. If this fails due to security restrictions, the editor tries again in VSAM read-only mode. If this also fails, you are notified and the ISET is not accessed.

In read-only mode, only some of the editor functions are available. Functions that physically alter data in the ISET are not permitted.

Important! When you are editing an IMOD in read-only mode, you will receive a cautionary message as you start to edit the IMOD. However, you will not be able to save the IMOD that you have edited. To save changes you have made to the IMOD, use the ISPF CREATE or REPLACE command to save the IMOD temporarily in a sequential data set. When you have update access to the ISET, you can re-edit the IMOD and copy the data back from the sequential file.

IMOD Naming Conventions

IMOD names can be up to 16 characters in length. The following restrictions apply:

- An IMOD name must not start with a dollar sign (\$). CA reserves names beginning with a dollar sign for their use.
- An IMOD name should use only characters supported for standard REXX labels. For example, if you name an IMOD ABC.XYZ, REXX will not support the statement, CALL ABC.XYZ; the IMOD name is treated as a stem variable instead.
- The underscore character (_) has special meaning in an IMOD name. It is used for name prefixing.

Name Prefixing

When creating IMOD names, you should group them into logical units by using name prefixes. The prefix can be any character string (except those beginning with \$) and can end with an optional separator. The resulting IMOD name must obey the REXX rules for labels.

Except for IMODs invoked as subroutines or functions, underscores in IMOD name prefixes have a special use. When an IMOD task is created and the requested IMOD does not exist or is not usable, the name is shortened from the right. The last character and then all preceding characters up to, but not including, an underscore are discarded. The shortened name is then used to search for an executable IMOD. If one is still not found, the shortening process is repeated until an executable IMOD is found or there are no characters left in the name. In this manner, you can provide default IMODs to be executed in place of the one requested.

Example

You define an operator console command, HELP. To use the HELP command, the operator enters `HELP xxx operands`, where `xxx` is the name of an IMOD of the form `HELP_xxx`. You also provide the following IMODs: `HELP_DISPLAY`, `HELP_VARY`, and `HELP_START`. If the operator enters `HELP VARY OFFLINE`, the `HELP_VARY` IMOD is executed, which then analyzes the operand `OFFLINE` and provides an appropriate response.

However, if the operator enters `HELP OFFLINE`, `ISERVE` will try to execute the `HELP_OFFLINE` IMOD, which does not exist. `ISERVE` then shortens the IMOD name to `HELP_`. If you have provided an IMOD of that name, it is executed. In this example, it might be sufficient for the `HELP_` IMOD to send the message, `HELP NOT AVAILABLE`, back to the operator.

IMOD Editor

The IMOD editor is used to edit existing IMODs and to write new ones. Although IMODs are stored in VSAM-based ISETs, the IMOD editor uses ISPF edit and browse commands to operate like other ISPF applications. It supports scrolling, program function (PF) keys, and fast-path commands.

The IMOD editor provides the following panels from which you can create, display, modify, and execute IMODs:

- ISET panel
- IMOD panel
- IMOD EDIT and BROWSE panels
- IMOD EXECUTE panel

Macros

You can use the following macros on the EDIT panel:

ERROR

Searches forward from the current line for the next compiler error message. The panel is then repositioned to this line.

Note: When you access the EDIT panel for a compiled IMOD that has error messages, the ERROR macro is invoked automatically.

ICOPY *imodname* [*iset*]

Copies the text from another IMOD to the one being edited.

imodname

Specifies the name of the IMOD whose text you want to copy.

iset

(Optional) Specifies the name of the ISET that contains the IMOD to be copied.

Because the actions of ICOPY are carried out through the primary ISERVE address space, CA-GSS must be running for it to work.

SRVEINT

Is invoked at EDIT panel initialization to establish the editing environment. You can modify this macro, which is in the CAICLS0 data set.

SRVHELP1

Inserts the function prototype above the cursor position when the cursor is on a valid function name (the name must end with a left parenthesis).

You should assign this macro to a PF key.

Data Integrity

The IMOD editor permits users on multiple systems to share ISETs. ISETs and IMODs are protected by global enqueues. For this protection to be effective, all systems having access to the ISET must be members of a global resource serialization (GRS) ring or equivalent multi-image manager.

When an IMOD is being edited by a user, other users can still browse the last-saved copy.

When an ISET is being updated, an exclusive enqueue is obtained for QNAME IPGMGREX and RNAME F.*dsn* (*dsn* is the 44-byte cluster name, right-padded with blanks).

When an IMOD is being updated, an exclusive enqueue is obtained for QNAME IPGMGREX and RNAME P.*imod.dsn* (*imod* is the 16-byte IMOD name, right-padded with blanks; *dsn* is the 44-byte cluster name, right-padded with blanks).

IMOD Source Recovery

If the system fails while you are processing a command that involves updating, you can recover your IMOD source by retrieving a temporary copy of the original.

Before updating, the IMOD being updated is saved under a temporary name. If the operation completes, this temporary copy is deleted. If the system fails, the temporary copy is retained for you to retrieve. The following table defines the names of the temporary copies:

Update From	Temporary Copy Name	Variable Contents
TSO	\$\$\$\$SAV_ <i>userid</i>	Your TSO user ID
Batch	\$\$\$\$SAVB_ <i>jobname</i>	The job name of the batch job performing the update

You can browse, rename, or delete a temporary copy, but you cannot edit or compile it.

REXX Language in CA-GSS

Note: CA-GSS supplies extended functions for you to use when writing IMODs. For information about these functions, see the *Reference Guide*.

A REXX program is built from a series of instructions, commands, and function calls:

Instruction

Contains one or more clauses. In many instructions, the first clause starts with a keyword that identifies the instruction. Some instructions affect the flow of control, while other instructions provide services to the programmer. Some instructions, such as DO, include nested instructions.

Function

Is a built-in subroutine that processes data and returns a value to the IMOD.

Command

Is any text string that is not an instruction, function, or operand. Commands are passed to the current environment for evaluation and disposition. The environment is determined by the program using the ADDRESS instruction and can be any of several non-REXX programs. For example, commands can be passed to the CA Jobtrac Job Management environment for processing as CA Jobtrac Job Management commands.

CA-GSS supports all REXX instructions, commands, and functions, with the following exceptions:

- The INTERPRET command is not supported.
- Labels cannot be duplicated.
- Function names and labels to be called are limited to 32 characters.

The REXX language uses the following conventions:

- Variables—Under REXX, all manipulated information must be stored as character strings in variables.
- Argument strings—When a REXX program or subroutine is invoked, one or more text strings can be passed to it as argument strings. How these strings are subdivided or parsed is up to the programmer.
- Result string—After a REXX program completed, it can pass a character string (result) back to the calling program.
- Program stack—The stack is a sequential collection of records of any content or length that is passed to a REXX program from an external source. This external source can either invoke the REXX program or be a program invoked from REXX. For this reason, the stack is sometimes referred to as the external data queue (EDQ). The REXX program can itself use the stack for working storage, although data must be moved to variables before most manipulations.

Commands processed by external environments frequently accept and return data on the stack.

- Internal subroutines—A REXX program can contain subroutines. Such subroutines can access variables that belong to the calling routine. Internal subroutines are always compiled with the main routine, and are considered a part of the main routine.
- External subroutines—A REXX program can issue calls to external subroutines. External subroutines are separate from the routine that issues the call. An external subroutine does not share any variables with its caller except for global variables. The only values that can be passed are arguments, a result string, and the stack.

Binary Conversion in REXX

Because REXX deals only with character strings and not with the binary values used by IBM computers, it is essential that binary data be converted to a character string. The following statement converts a binary value *i* to a character string:

```
i = C2D(i).
```

Before storing numeric values in memory, you may need to reconvert them back to binary values. The following statement converts a character string *i* to a 4- byte binary value:

```
i = D2C(i,4).
```

Example: Binary to Character Conversion

REXX treats the full word X'00000024' as a character string of length 4. If you try to print it, you get a 4-byte unprintable string. If you try to use it numerically, you get a data conversion error. However, where *i* = X'00000024', coding *i* = C2D(*i*) converts *i* to its decimal equivalent of 36. REXX can perform arithmetic functions on this value.

Converting Back to Binary Values

Prior to storing numeric values in memory, you may need to reconvert back to binary values. To return a value *i* to a binary value, code:

```
i = D2C(i,4).
```

Convert Back to Binary Values

Prior to storing numeric values in memory, you may need to reconvert back to binary values. To return a value *i* to a binary value, code:

```
i = D2C(i,4)
```

ADDRESS Environments

ADDRESS environments let you communicate with other products and programs external to ISERVE. They are established by issuing an ADDRESS instruction for a specific environment. They have the following format:

ADDRESS *environment command*

environment

Specifies the name of a product or product component.

command

Specifies a valid command for that product. All REXX-defined ways of specifying ADDRESS are supported.

ADDRESS environments are available for the following components or products:

- ISERVE
- IDCAMS
- CA Jobtrac Job Management
- CA MIA Tape Sharing
- CA OPS/MVS Event Management and Automation
- CA SYSVIEW Performance Management
- CA View

Other CA Technologies products may provide their own ADDRESS environments. Some non-CA Technologies products provide TSO/E REXX-compatible ADDRESS environments that can also be used with CA-GSS.

ADDRESS Environment for ISERVE

The ISERVE ADDRESS environment is the initial default command environment for IMODs. It is the required environment to process all TSO/E stack command emulations. You do not need to specify ADDRESS ISERVE unless you change the default environment by coding ADDRESS *environment* without the command.

Example: ISERVE Command in an IDCAMS ADDRESS Environment

The ADDRESS IDCAMS instruction changes the default environment from ISERVE to IDCAMS. The ADDRESS ISERVE 'NEWSTACK' instruction issues a stack command to the ISERVE environment but does not change the default from IDCAMS.

```
'DELSTACK'          /* Clear STACK from default
                    ISERVE environment */
ADDRESS IDCAMS      /* Change default environment to
                    IDCAMS */
'LISTCAT' dsname1   /* Enter command to IDCAMS */
'LISTCAT' dsname2   /* Enter command to IDCAMS */
ADDRESS ISERVE 'NEWSTACK' /* Enter STACK command to the
                    ISERVE environment */
```

To change the default back to ISERVE, you specify the ADDRESS ISERVE instruction.

ADDRESS Environment for IDCAMS

The ADDRESS environment for IDCAMS executes any IDCAMS command.

Note: For a complete description of IDCAMS commands, see IBM's *DFSMS Access Method Services for Catalogs*.

Each time you enter a command in the IDCAMS environment, you invoke a new copy of IDCAMS. For this reason, information cannot be carried from one command to the next. You can conditionally execute commands based on IDCAMS return codes only when you specify those commands in REXX.

An input command can be up to 32,760 characters. Input to IDCAMS is case-sensitive.

The instruction to establish an ADDRESS environment for IDCAMS has the following format, where *command* is any IDCAMS command:

```
ADDRESS IDCAMS command
```

Output lines are returned on the stack and are identical to those that would be produced in the SYSPRINT data set, including ASA carriage control characters and page headers.

Important! Because IDCAMS can have voluminous output, avoid memory depletion by limiting the amount of data returned on the stack.

ADDRESS Environment for CA Jobtrac Job Management

An ADDRESS environment is available for interacting with CA Jobtrac Job Management, a CA Technologies automated scheduling product. This ADDRESS environment lets you obtain information from the CA Jobtrac Job Management checkpoint data set from IMODs.

ADDRESS Environment for CA MIA Tape Sharing

The Tape Preferencing and Control Facility (TPCF) of CA MIA Tape Sharing provides an API. This API is available in an ADDRESS environment as ADDRESS TPCF.

Using this ADDRESS instruction, you can pass a single device address and receive a stack record describing the status for that device.

The returned data contains bit-level flags and binary fields. To enhance the availability of ADDRESS TPCF, CA-GSS provides a service routine called \$SRV_TPCF. You can call this service routine as an external subroutine from an IMOD. The invoking argument consists of a device address, several device addresses (separated by commas), a range of device addresses (separated by hyphens), or a combination of the preceding. The results are returned on the stack, one device per line, with all fields converted to standard REXX-type display and computational values.

Note: For a complete description of \$SRV_TPCF, see the *Reference Guide*.

ADDRESS Environments for CA OPS/MVS Event Management and Automation

Standard REXX ADDRESS environments are available for interacting with CA OPS/MVS Event Management and Automation. The following environments are available:

Environment	Description
AOF	Enable or disable a CA OPS/MVS Event Management and Automation rule (results returned).
OPER	Issue a console operator command and receive a reply.
OPSREQ	Run a CA OPS/MVS Event Management and Automation request rule (no results returned).
OSF	Run a command in a CA OPS/MVS Event Management and Automation server address space (no results returned).

Communication Between IMODs and CA OPS/MVS Event Management and Automation

Communications between CA OPS/MVS Event Management and Automation and ISERVE are multi-threaded, so there is no limit to the number of requests that can be processed simultaneously from concurrently executing IMOD tasks. However, only one version of the CA OPS/MVS Event Management and Automation ADDRESS environment can be available to a single ISERVE address space. Therefore, you cannot simultaneously test two separate versions of CA OPS/MVS Event Management and Automation.

By default, communication with CA OPS/MVS Event Management and Automation is to the system using the subsystem ID defined in the CA-GSS initialization parameters. You can override this value through the SETADDR() function.

Note: For information about the SETADDR() function, see the *Reference Guide*.

ADDRESS Environment for CA SYSVIEW Performance Management

You can access the CA SYSVIEW Performance Management API using REXX to obtain information from CA SYSVIEW Performance Management displays for use in other programs.

Note: For more information, see the CA SYSVIEW Performance Management documentation.

ADDRESS Environment for CA View

An ADDRESS environment is available for interacting with CA View, a CA Technologies automated report distribution product. This ADDRESS environment lets you execute CA View commands from IMODs.

Note: For more information, see the CA View documentation.

Extended Return Codes

Unlike standard REXX, IMOD extended functions return a numeric return code in the REXX RC special variable. When programming in REXX, you should check the RC special variable for proper completion. For example, the SAM(GET ...) function can return the value ARG 1 MISSING OR INVALID. If the value of the RC special variable is 0, no error occurred (that is, the result was read from a disk file and is a valid record).

Note: IMODs called as external subroutines can set the RC special variable by issuing the SETRC() function. You can trap errors from extended functions by using these instructions:

- SIGNAL ON ERROR
- SIGNAL ON FAILURE
- CALL ON ERROR
- CALL ON FAILURE

The following table shows the general meanings for return codes:

If the return code is	Then the function or subroutine has
0	Completed successfully
Greater than 0	Not completed successfully (SIGNAL ON ERROR)
Less than 0	Not completed successfully because of a serious failure (SIGNAL ON FAILURE)

Error return codes in the range -100 through 120 are explained in the following table. All other error return codes are function-dependent.

A return code of	Indicates
-100	An unexpected abend (for example, 0C4 from the WTO() function).
-99 through 99	Systems applications architecture (SAA) REXX return codes. Text for these error codes can be obtained from the ERRORTXT() REXX built-in function. Note: For a list of the SAA error codes and their descriptions, see IBM's REXX documentation.
100	An expected abend (for example, 0C4 from the MEMORY() function).
101 through 120	A missing or invalid argument specification corresponding to arguments 1 through 20 respectively.

Program Stacks

A *program stack*, also known as an external data queue (EDQ), is created for each IMOD task. Stacks are maintained until the IMOD task has completed. Any data remaining on the stacks is discarded or returned to the caller.

Stacks are common to all programs called from an IMOD task and are accessible to ADDRESS environments. The SPAWN() function can pass the current stack (or a copy of it) to a newly created IMOD task.

A stack consists of a series of records that can contain data in any format. In ISERVE, the length of a single record cannot exceed 32,000 characters. The maximum number of records is determined by the amount of free memory in the ISERVE address space.

To prevent programming errors from seriously degrading ISERVE performance, each stack is limited in the amount of data it can contain. By default, an ISERVE stack can contain up to one megabyte (1,000 KB) of data. You can change the default value through the MAXSTACK initialization parameter. The maximum size of any stack can be controlled by the IMOD through the STACKINF() function.

REXX-defined Stacks

The basic REXX implementation provides an IMOD task with a single stack. Conceptually, a stack has a top and a bottom. Records can be added at the top (using PUSH) or at the bottom (using QUEUE). The number of records on a stack can be determined with the QUEUED() built-in function. Records can only be removed from the top of the stack with the PULL and PARSE PULL instructions.

TSO/E Stack Emulation

In addition to the stack instructions defined by the language (such as PULL and QUEUE), TSO/E provides additional commands. The following list shows the ISERVE extensions that emulate these commands.

Important! Although ISERVE supports these extensions, it is recommended that you do not use them. Instead, use the ISERVE stack functions such as SWAPSTAK() and PUBSTACK(), which are easier to use.

DELSTACK

Deletes all records in the current stack and removes the last added NEWSTACK barrier. Records that existed on the stack before the last issuance of the NEWSTACK command are accessible again.

DROPBUF

Deletes all records in the current stack buffer, as well as the buffer itself. With the DROPBUF command, you can specify a single numeric value that deletes all buffers up to and including the buffer identified by that number. For example, you can enter DROPBUF 0 to delete all records and all buffers. (DROPBUF 0 is equivalent to DELSTACK.)

MAKEBUF

Divides a stack into buffers. MAKEBUF places a barrier on top of the current stack. The PUSH and QUEUE commands operate as if nothing exists below this barrier. However, both the QUEUED() function and the PULL command ignore buffer barriers. When a PULL crosses a buffer barrier, the buffer count is reduced by one and the buffer is discarded.

NEWSTACK

Constructs a barrier at the top of the stack. Records below this barrier cannot be accessed until a corresponding DELSTACK is issued. In essence, NEWSTACK hides the existing stack and starts a new one.

QBUF

Returns the current buffer number in the RC special variable. The lowest buffer number is zero.

QELEM

Returns the total number of stack elements in the current buffer. The value is returned in the RC special variable.

QSTACK

Returns the current stack number (as determined by the nesting level of NEWSTACK and DELSTACK operations) in the RC special variable. The lowest stack number is zero. An RC value equal to zero indicates that you are currently processing the only stack left.

ISERVE Stack Extensions and Functions

To facilitate the use of stacks, ISERVE provides extensions that are accessible through extended functions. These extensions do the following:

- Provide each IMOD with multiple, fully addressable stacks. An IMOD can use from 1 to 230 stacks and switch freely between them. Unlike the NEWSTACK and DELSTACK commands, you do not need to delete the current stack to get to a previous one.
- Let stacks receive TRACE and SAY output, making the output available to the program or program caller.
- Permit access to a stack in random fashion, so that records can be read or written in any order.
- Permit an IMOD task access to stacks that belong to other executing IMOD tasks. Coupled with the WAIT() function, this enables communication between concurrently executing IMOD tasks.

The following table lists the ISERVE stack functions.

Note: For more information about the stack functions, see the *Reference Guide*.

PUBSTACK()

Controls the use of a stack by external IMODs.

PULL()

Operates the same way as the PULL instruction in REXX, except that you can specify the stack number and the owning IMOD task ID.

PUSH()

Operates the same way as the PUSH instruction in REXX, except that you can specify the stack number and the owning IMOD task ID.

QUEUE()

Operates the same way as the QUEUE instruction in REXX, except that you can specify the stack number and the owning IMOD task ID.

QUEUES()

Operates the same way as the QUEUE() function, except that the entire contents of a stack are queued in a single operation.

REDIRECT()

Sends TRACE or SAY output to a stack.

SHOVE()

Inserts a record anywhere in any stack.

SORT()

Lets you sort stack records based on the contents of up to six fields.

STACKINF()

Returns information about stacks, including the IDs of all active stacks, and lets you change the maximum permissible size of a stack.

SWAPSTAK()

Controls which stack is currently in use.

TUG()

Enables reference to any record in any stack. TUG() obtains a copy of the record only; the record is not removed from the stack.

WAIT()

Lets an IMOD suspend processing. Processing is resumed only when an external IMOD writes one or more records to the waited-upon stack or stacks.

YANK()

Enables reference to any record in any stack. It operates the same way as TUG(), except that the record is removed from the stack.

Stack Record Identifiers

Each stack record that is written from another IMOD task is tagged with control information. The originating GoalNet node, associated user ID, security group, and IMOD ID are not retrievable by standard REXX instructions such as PULL. You can retrieve this information by using the stack functions provided by ISERVE.

SYSIN and Stacks in Batch Mode

In batch mode (both under SRVBATCH and in packaged IMODs), stacks function in the same manner as in online mode with one exception. If PULL or PULL() is issued against an empty stack, records are fetched from the file defined by the SYSIN DD statement.

Work Stacks

When using stacks, it is sometimes useful for a subroutine to be able to obtain a new work stack, use it, and then delete it, returning the environment to exactly the state it was in when the subroutine was called.

Example: Work Stack Usage

The work stack is made the current stack so that REXX instructions such as PUSH, PULL, QUEUE, and so on can operate on it. Access to other stacks (those created by the calling IMOD) can be gained by using the extended functions, PUSH(), PULL(), QUEUE(), and so on.

```
entry_pt:
  orig = swapstak()      /* obtain existing current stack id */
  work = swapstak('new') /* obtain an empty stack; make it   */
                        /* the current stack                */

  processing ...
  ...
  x = swapstak('delete') /* eliminate work stack             */
  x = swapstak(orig)     /* return environment to the        */
                        /* original stack                    */

  return
```

GoalNet

Note: For information about how to set up GoalNet, see the *Installation Guide*.

You use the CALLX() function to execute an IMOD at a remote GoalNet node.

Note: For information about the CALLX() function, see the *Reference Guide*.

Using GoalNet resembles a call to a REXX external subroutine, except that you specify the node where the subroutine is to execute. You can pass arguments and the contents of the current stack to the called routine. Following completion of the routine, the resulting stack and the return string are returned to the caller.

GoalNet Log

Upon initialization of GoalNet, a log data set is opened to record GoalNet-related information. This data set is either defined by the GNETLOG DD statement in the CA-GSS started task or a dynamically allocated SYSOUT file.

ILOG Files

ILOG files are log files used for logging IMOD messages and events. They are memory-mapped VSAM linear data sets (LDSs) that can be used to store and retrieve information.

How ILOG Files Operate

Each ILOG file is made up of one to ten subfiles. Only one subfile is in use at a time. Switching recording to another subfile prevents further writing on the previous subfile.

When an ILOG subfile is full, CA-GSS marks it “dump required,” and the next available subfile is cleared and used for recording. When you no longer need the data on the filled subfile, you must reset it with the RESET ILOG operator command. The data in the reset subfile, however, remains intact until the subfile is actually needed for re-use.

Note: For information about the RESET ILOG operator command, see the *Reference Guide*.

Whenever an ILOG subfile is full and a switch occurs, the \$USER_ILOG_FULL special purpose IMOD is scheduled (if it is present). The argument string passed to this IMOD consists of the ILOG file number and subfile number. If this IMOD returns the result string RESET, the ILOG subfile is reset and made available for recording. Any other result string leaves the ILOG subfile in “dump required” status. If \$USER_ILOG_FULL is not available, the ILOG subfile is always reset.

Allocate ILOG Files

You allocate ILOG files as VSAM LDSs by using either the SRVMAINT utility (recommended) or IDCAMS.

To allocate ILOG files Using SRVMAINT

1. Determine the number of ILOG files required by the CA Technologies product.
2. Create a job using the following JCL statements:

```
//STEP1 EXEC PGM=SRVMAINT
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
ALLOC_ILOG -
  NAME xxx.xxx.ilog_n -
  VOLSER volser -
  TRK 5 5
...
//
```

Note: For information about the ALLOC_ILOG batch maintenance command, see the *Reference Guide*.

A job to allocate the required number of ILOG files is created.

3. Submit the job.

The ILOG files are allocated.

Define ILOG Files to CA-GSS

After you allocate the ILOG files, you must define them to the appropriate ISERVE address space before they can be accessed.

ILOG files cannot be shared by multiple ISERVE address spaces.

To define ILOG files to an ISERVE address space, add ILOG statements in its initialization parameter data set member, which is typically CAW0OPTN(RUNPARAM).

Important! The size of each ILOG file is determined by a value in the initialization parameter. Specifying large values consumes large amounts of virtual storage (and, when searched, large amounts of real storage). You should specify only enough storage to store the data you need.

Note: For information about the ILOG CA-GSS initialization parameter, see the *Reference Guide*.

ILOG Records Retrieval

The following functions let IMODs retrieve records from ILOG subfiles:

Function	Treats ILOG subfiles as	Which is useful for
ILOG()	Individual files. You must specify both the ILOG file number and the specific subfile you want to retrieve.	Copying a single subfile for archiving purposes.
ILOGG()	Parts of a single file in chronological order.	Searching for all occurrences of a particular message.

Note: For more information about the ILOG() and ILOGG() functions, see the *Reference Guide*.

ILOG Records Logging of Specific Data

The ILOGR() function lets IMODs write to an ILOG file directly.

Note: For information about the ILOGR() function, see the *Reference Guide*.

ILOG Records Logging of WTO Messages

The following actions log WTO messages issued from a specific address space to a specific ILOG:

- A MONITOR operator command is issued.
Note: For information about the MONITOR operator command, see the *Reference Guide*.
- A SCRASID() function is invoked.
Note: For information about the SCRASID() function, see the *Reference Guide*.
- An internal request is made by CA Insight Database Performance Monitor for DB2 for z/OS.

You can also use the IROUTE() function to specify the ILOG file where you want WTO text to be recorded.

Note: For information about the IROUTE() function, see the *Reference Guide*.

How IMOD Variables Are Accessed

REXX provides the PROCEDURE EXPOSE instruction to grant an internal subroutine access to the variables of a caller. The VVALUE() CA-GSS extended function grants an external subroutine access to the variables of its caller. Access is granted at all nesting levels of the calling chain, and variables can be read, updated, and created. An IMOD task can establish a variable pool, which is accessible to all external subroutines.

Predefined Variables

The following predefined variables are available to IMODs. Predefined variables have their values already assigned when an IMOD is entered. All predefined variables are reserved. If you change the value of a predefined variable, the new value is not passed to external subroutines.

imod_ascb

Contains the address of the address space control block (ASCB) responsible for triggering this IMOD. The value is an integer.

imod_asid

Contains the ASID of the address space responsible for triggering this IMOD. The value is an integer.

imod_console

Contains the ID of the console responsible for triggering this IMOD. The value is a positive integer. It is zero if the IMOD was not triggered by an operator command.

imod_group

Identifies the security group assigned to the task that triggered this IMOD.

imod_id

Contains the ID assigned to the IMOD. The value is an integer. It is used for external stack references and by the CANCEL CA-GSS/ISERVE operator command.

imod_ilog

Identifies the ILOG file that is the destination for the triggering WTO text. The value is blank if it is not applicable or if logging is not requested.

imod_jobid

Contains the ID of the job responsible for triggering this IMOD (for example, JOB 4321). The value is blank if the IMOD was not triggered by a WTO message.

imod_jobname

Contains the name of the job responsible for triggering this IMOD. The value is blank if the IMOD was not triggered by a WTO message.

imod_level

Contains the nesting level of external IMOD subroutine calls.

imod_lines

Contains the number of lines in the WTO message responsible for triggering this IMOD. The value is zero if the IMOD was not triggered by a WTO message.

imod_msg.n

Contains the text of the *n*th line of the WTO message that triggered the IMOD (for example, *imod_msg.1* is the text of the first message line). This information is also present as REXX arguments.

imod_node

Contains the name of the GoalNet node where this IMOD is processing.

imod_oursmfid

Contains the SMF ID of the system where ISERVE is executing.

imod_oursysname

Contains the system name of the system where ISERVE is executing.

imod_replyid

Contains the reply ID of the WTOR message responsible for triggering this IMOD. The value is a null string if the IMOD was not triggered by a WTOR message.

imod_sysname

Contains the name of the GRS system where the triggering WTO message was issued (usually, the system where CA-GSS is executing). The value is blank if this IMOD was not triggered by a WTO message.

imod_timestamp

Contains the timestamp, in *hh.mm.ss* format, of the triggering WTO message. The value is blank if the IMOD was not triggered by a WTO message.

imod_userid

Contains the user ID assigned to the task that triggered the IMOD.

Global Variables

Global variables are REXX variables that begin with an ampersand (&). One copy of each global variable is maintained in an ISERVE address space, and all IMODs executing in this address space have access to it. In this way, information can be shared by multiple IMODs and multiple executions of the same IMOD. Global variables are initialized using the GLOBVAL initialization parameter.

Note: An uninitialized global variable has a null value, rather than being equal to the variable name.

Global variables have the following restrictions:

- Your global variable names may not begin with the character sequence &\$ (ampersand dollar sign). These characters are used by IMODs supplied by CA.
- The DROP instruction does not release the storage of a global variable. The storage is re-used if the same global variable is re-assigned.
- Global variables can contain up to 80 characters of data. If you require more characters, you must pre-allocate it by using the GLOBVAL initialization parameter or the \$GLOBAL() function.

Note: For information about the GLOBVAL initialization parameter and the \$GLOBAL() function, see the *Reference Guide*.

External Subroutines

External subroutines are IMODs that are called as subroutines or invoked as functions by other IMODs. Parameters are passed and returned according to the rules of REXX. You can nest any combination of external subroutines or functions up to 50 levels at a single GoalNet node.

Important! To be called as an external subroutine, an IMOD *must* contain the compiler directive #CALLABLE.

You can use the \$WHEREFROM() function to identify the IMOD and statement number that made the external subroutine call.

Compiler Directives

Your IMOD can contain one or more of the following compiler directives. These directives control aspects of IMOD use and execution. They must begin with the pound sign (#) in Column 1 and the first letter of the directive in Column 2.

#DESC

Describes the IMOD.

#CALLABLE

Specifies that this IMOD can be called from another IMOD as an external subroutine.

#SOURCE

Loads the source statements and object code into memory.

#REFORMAT

Reformats the source IMOD.

Note: For more information about compiler directives, see the *Reference Guide*.

User-defined Functions and ADDRESS Environments

CA-GSS provides an assembler language interface so that you can add your own functions and ADDRESS environments. Both interfaces are compatible with TSO/E REXX, and routines used in that environment are usually transportable to the CA-GSS service.

Under CA-GSS, functions and ADDRESS environments have few differences. The most obvious difference is in the way you invokes the routine. Other differences include the following:

- Functions can accept up to 20 arguments and must return a result.
- ADDRESS environments accept a single command string and can return a numeric value in the RC special variable.

Functions and ADDRESS environments can read and write stack records and access REXX variables. Under CA-GSS, functions and ADDRESS environments can execute under a subtask. Functions can also execute synchronously, provided they do nothing that would cause a wait (for example, performing I/O).

When functions and ADDRESS environments are called, Register 0 contains the address of the environment block. The address of the External Routines Table is in this block and is mapped by IBM's IRXEXTE macro. The External Routines Table contains the addresses of the service routines supported by CA-GSS.

User ID Routine

You can obtain the user ID assigned to the executing IMOD by using the USERID_ROUTINE vector. The parameter list is identical to that for the IRXUID routine supported by TSO/E REXX.

Data Stack Routine

You can access the stack by using the STACK_ROUTINE vector. The parameter list is identical to that for the IRXSTK routine supported by TSO/E REXX. Under CA-GSS, access is to the current stack.

Variable Access Routine

You can fetch values from and store values in REXX variables by using the IRXEXCOM vector.

SAY Instruction Routine

You can produce a SAY output by using the IRXSAY vector. There is no difference between the WRITE and WRITERR functions. Under CA-GSS, SAY output is treated in the same manner as that produced by the IMOD. By default, SAY output is printed in ISRVLOG. The IMOD can redirect SAY output to a stack (using the REDIRECT() function) or to a SYSOUT data set (using the SAYWHAT() function).

During function and ADDRESS execution, SAY output is accumulated in a work area. At the conclusion of the function or ADDRESS, the accumulated data is sent to the appropriate destination. If an abend occurs, the accumulated data may be lost.

Function Arguments

Each function is passed the address of a parameter list, mapped by the IBM's IRXEFPL macro. The parameter list is identical to the one supported by TSO/E REXX.

Function Return Codes

Upon completion, functions must return a zero value in Register 15. Any other value may cause an error condition to be raised in the IMOD.

ADDRESS Environment Arguments

ADDRESS environments are passed a parameter list in Register 1. This list consists of a five-word list of addresses. Each address points to a parameter. The parameters are as follows:

- Eight-byte name of the host command environment
- Four-byte address of the command string to be processed
- Four-byte length of the command string
- Reserved
- Four-byte area in which you will place a value to be set in the RC special variable

ADDRESS Environment Return Codes

CA-GSS treats return codes differently from TSO/E REXX. Under TSO/E REXX, the return code from the command is placed in a parameter passed to the routine, and the return code from the host environment (which may be set up to process multiple commands) is returned in Register 15. The value in the parameter word is used to set the RC special variable, and the value Register 15 is used to control continued execution of the EXEC.

Under CA-GSS, the following convention is used:

- If Register 15 is zero, the RC special variable is set from the parameter list value.
- If Register 15 is not zero and the parameter list value is zero, the RC special variable is set from Register 15.
- If both Register 15 and the parameter list contain non-zero values, the parameter list value takes precedence.

Coding Requirements

Functions and ADDRESS environments must be link edited, and they must execute AMODE 31 and reside in an APF-authorized library. RMODE ANY is desirable but not required. Debugged routines should be reentrant.

Function Coding

When you create a function, you should consider the following:

- Functions cannot preserve data (except in REXX variables) across calls.
- You identify your function to CA-GSS using the FUNCTION initialization parameter. You can specify the interface type to be either SYNC or ASYNC:
 - If you specify SYNC, the function is invoked as a called subroutine. It must not do anything that would release control to the operating system (such as I/O, WAIT, and some SVCs). This is the most efficient interface and is appropriate for functions that manipulate data already in memory.
 - If you specify ASYNC, the function is invoked as a subtask, and all processing occurs asynchronously. This isolates other IMODs from waits for I/O, SVCs, and so on.
- When the function is processed under a subtask, it still executes under the authority of the user ID assigned to the IMOD.
- If you access the stack, the stack records are returned to your function in protected storage. If you want to modify the data, you must make a copy in your own storage area.

ADDRESS Environment Coding

When you create an ADDRESS environment, you should consider the following:

- ADDRESS environments cannot preserve data (except in REXX variables) across calls.
- You identify your ADDRESS environment to CA-GSS using the ADDRESS initialization parameter. Under CA-GSS, all ADDRESS environments execute in subtasks. The first time an IMOD task issues a particular ADDRESS command, a subtask for that environment is created and assigned to the IMOD task for its exclusive use. Repeated ADDRESS commands from the same IMOD task for the same environment reuse the same subtask. When the IMOD task completes, the subtask is retained and reassigned to the next IMOD task that uses the ADDRESS environment.

You can prevent other IMOD tasks reusing the subtask by specifying DETACH on the ADDRESS initialization parameter. This destroys the subtask when the IMOD task completes.
- An ADDRESS environment subtask executes under the authority of the user ID assigned to the IMOD issuing the ADDRESS command.
- If you access the stack, the stack records are returned to your ADDRESS environment in protected storage. If you want to modify the data, you must make a copy in your own storage area.

Example: Function

A sample function that can be used under TSO/E REXX and CA-GSS is provided in the CAW0JCL(BYSSFUN) data set member. When executed, it takes the following actions:

- Issue the following as SAY output:
 - User ID
 - Argument strings, if any
 - Stack contents, if any
- Place the string, NEW VALUE OF "SAMPLE" VARIABLE, in the SAMPLE variable.
- Return the string, THIS IS THE RETURNED RESULT STRING.

You can use the following REXX EXEC to check the operation of the BYSSFUN function:

```
queue 'stack_record_1'
queue 'stack_record_2'
queue 'stack_record_3'
rslt = BYSSFUN('arg_1',, 'arg_3')
say 'returned string:' rslt
say '"sample" variable:' sample
```

Example: ADDRESS Environment

You can install ADDRESS environments in the IBM-prescribed manner. However, the CAW0JCL(BYSADDA) data set member is provided for you to create a sample ADDRESS environment that can be used under TSO/E REXX and CA-GSS. When assembled and link edited, this member is a TSO/E REXX function that dynamically adds an ADDRESS environment to TSO/E REXX:

```
address subcom 'samppadr' /* test for existence of environment */
if rc ^= 0 then do
  rslt = addaddr('BYSSADD', 'BYSSADD') /* add environment */
  say 'addaddr returns' rslt
end
```

When executed, the ADDRESS environment takes the following actions:

- Issue the following as SAY output:
 - User ID
 - Command string, if any
 - Stack contents, if any.
- Place the string, NEW VALUE OF "SAMPLE" VARIABLE, in the SAMPLE variable.
- Return the value, 999, in the RC special variable.

You can use the following REXX exec to check the operation of the BYSSADD ADDRESS environment:

```
queue 'stack_record_1'  
queue 'stack_record_2'  
queue 'stack_record_3'  
address BYSSADD 'command_string'  
say 'rc variable:' rc  
say "sample" variable:' sample
```

How IMODs Are Created

The process to create IMODs is as follows:

1. Allocate an ISET as a container for the IMODs.
2. Define the ISET to CA-GSS to make it accessible to the IMOD editor.
3. Write your IMODs.
4. Compile your IMODs.
5. Load the IMOD in a CA-GSS address space.
6. Test the IMOD.

Allocate New ISET Data Sets

IMODs are contained in ISET VSAM KSDSs. The source and compiled object code, as well as control information, are included in these data sets.

You allocate ISET data sets by using either the SRVMAINT utility (recommended) or IDCAMS.

To allocate a new ISET data set

1. Create a job using the following JCL statements:

```
//STEP1 EXEC PGM=SRVMAINT
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  ALLOC_ISET -
  NAME xxx.xxx.imod -
  VOLSER volser -
  CYL 2 1
//
```

A job to allocate the ISET data set is created.

Note: For information about the ALLOC_ISET batch maintenance command, see the *Reference Guide*.

2. Submit the job.
The ISET data set is allocated.

Define an ISET to CA-GSS

After you allocate an ISET data set, you must define it to CA-GSS before it can be accessed by CA-GSS and the IMOD editor.

To define an ISET to CA-GSS, add an ISET initialization parameter in the CAW0OPTN(ISETS) member.

Note: For information about the ISET CA-GSS initialization parameter, see the *Reference Guide*.

The CAW0OPTN(ISETS) member is included in the following members:

- CAW0OPTN(RUNPARM), which makes the ISET available to CA-GSS
- CAW0OPTN(EDITPARM), which makes the ISET available to the IMOD editor

Write an IMOD

Important! Before you write an IMOD, you should become familiar with the REXX language (see IBM's guides) and the [CA extensions of REXX](#) (see page 296).

To write an IMOD

1. Ensure that the required ISET is defined.
2. Access the IMOD editor using *one* of the following methods:
 - Select the option from your ISPF menu.
 - Execute the GSSEDIT CLIST in the CAICLS0 data set.
 - Issue a command from a CA Technologies product (for example, CA Jobtrac Job Management and CA SYSVIEW Performance Management)

The list of defined ISETs appears.

3. Enter **S** beside the required ISET.

The list of IMODs in the ISET appears.

4. Enter the following command at the COMMAND ==>.prompt:

```
E imodname
```

imodname

Defines the name of the IMOD you want to write or edit.

The EDIT panel appears.

5. Write your IMOD using [compiler directive statements and REXX](#) (see page 296). Consider the following:
 - You should specify a #DESC compiler directive statement to describe the IMOD.
 - If the IMOD is an external routine, you must specify a #CALLABLE compiler directive statement.
 - You must not use the NUM ON command or enter line numbers manually. REXX treats line numbers as syntax errors.
 - You cannot use the COPY, REPLACE, APPEND, or CREATE commands to move material directly between ISETs. You can use these commands only when moving material between an IMOD and a sequential, non-ISET data set.

6. Press PF3.

The IMOD is saved in the selected ISET and is ready to be compiled.

Note: For more information about the IMOD editor panels, see the online help.

Compile an IMOD

Before you can load and execute an IMOD, you must compile it.

To compile an IMOD from the IMOD editor

1. Access the IMOD you want to compile.
A list containing the required IMOD appears.
2. Enter **C** in the input field beside the name of the IMOD.
The first character in the STATUS column field indicates the result of the compilation:
 - C indicates success.
 - * indicates error.
3. If the status indicates errors, enter **E** beside the IMOD.
The IMOD opens with the cursor at the first error.
 - a. Review the IMOD for errors, and correct them. To move through the errors, enter **FIND P'.** at the COMMAND ==> prompt. You can also use the ERROR macro by assigning it to a PF key.
 - b. Repeat Step 2 to recompile the IMOD.

You can also compile an IMOD in batch by using SRVMAINT and including the COMPILE batch maintenance command in your JCL job.

Note: For information about how to use the SRVMAINT batch maintenance program, see [Batch Maintenance of ISETs and ILOGs](#) (see page 333). For information about the COMPILE batch maintenance command, see the *Reference Guide*.

Load a Compiled IMOD

Before you can execute a compiled IMOD, you must load it in a CA-GSS address space.

To load a compiled IMOD from the IMOD editor

1. Access the compiled IMOD you want to load.
A list containing the required IMOD appears.
2. Enter **P** or **F** in the input field beside the name of the IMOD.
The IMOD is in the correct status for loading.

Note: For more information about the P and F command characters, see the online help.

You can also change the status of an IMOD to prepare it for loading by using SRVMaint and including the CHANGE batch maintenance command in your JCL job.

Note: For information about how to use the SRVMaint batch maintenance program, see [Batch Maintenance of ISETs and ILOGs](#) (see page 333). For information about the CHANGE batch maintenance command, see the *Reference Guide*.

3. Specify the subsystem ID of the target CA-GSS/ISERVE address space in the SSNAME field at the lower right of your panel, and enter L beside the IMOD.

The following message appears at the upper right of your panel:

IMOD Loaded

If the IMOD is already loaded, it is replaced.

Test a Loaded IMOD

Before moving an IMOD into production, you should test it to verify that the IMOD is working properly.

You test a loaded IMOD from the EXECUTE panel of the IMOD editor. From this panel, you can specify argument strings and the initial stack contents. After the execution is complete, you can retrieve the return string and any resulting stack contents.

To test a loaded IMOD

1. Enter X beside the loaded IMOD you want to test.

The EXECUTE panel appears.

Note: For information about the IMOD editor EXECUTE panel, see the online help.

2. Review and complete the panel as follows:
 - Complete the SSID and GoalNet Node (if required) fields to identify the CA-GSS address space where the IMOD is to execute. You can execute the IMOD in an address space that is active on the same CPU as your TSO session or an address space that can be reached using GoalNet.
 - Review the following fields to check that the default stack numbers satisfy your requirements. Update the fields as required.

Work STACK Number

Redirect SAY Output to STACK

Redirect TRACE Output to STACK

3. Specify any arguments for the IMOD as follows. All arguments are case-sensitive.
 - To pass a single, short argument string, specify it in the ARG field.
 - To pass multiple argument strings or a long argument, follow these steps:
 - a. Leave the ARG field blank, and enter **S** beside the Specify Alternate Arguments option.

The EDIT --- ARGUMENTS ISPF editor panel appears.
 - b. Specify the arguments.

The length and content of the individual arguments are governed by the ISPF editor. The aggregate length of all argument strings must be less than 4094, calculated as follows:

$$\text{number_arg_characters} + \text{number_arg_lines} \times 2$$
 - c. Press PF3.

The arguments are retained for the entire IMOD editor session and are passed to any executed IMOD whenever the ARG field is blank.
4. Specify any initial stack contents as follows. Stack records are case-sensitive.
 - a. Enter **S** beside the Load Initial STACK Values option.

The EDIT --- INITIAL STACK ISPF editor panel appears.
 - b. Specify the stack contents.

The stack that is passed to the IMOD can contain any records you want. The length and content of the individual stack records are governed by the ISPF editor. The aggregate length of all stack records, calculated as follows, must be less than 32766: $\text{number_stack_characters} + \text{number_stack_records} \times 2$
 - c. Press PF3.

The stack records are retained for your entire IMOD editor session and are passed to any executed IMOD.
5. Enter **S** beside the Execute the IMOD option.

Your request is passed to the specified address space. After an IMOD is executed, the result (as specified by the terminating RETURN or EXIT instruction) and the final stack contents are available for your inspection in the displayed EDIT --- REST RESULTS panel. If an error occurs, an ISPF error indication appears at the upper right of the panel.

IMOD Execution Using SRVBATCH

You can execute an IMOD in an isolated environment by using the SRVBATCH program. SRVBATCH is useful for the following:

- Debugging IMODs offline
- General REXX programming in batch

So that SRVBATCH accurately reflects how IMODs execute in an address space, CSA areas and structures are simulated in private storage. All subtasking, address environments, and so on, are provided in the same manner.

Considerations

When you use SRVBATCH to execute a IMOD, you should consider the following:

- You can execute compiled IMODs in batch mode.
- To use ADDRESS environments and external functions, you must define them using the ADDRESS and FUNCTION initialization parameters.
- Any COMMAND, PRODUCT, SSID, and WTO initialization parameters are syntax-checked, but otherwise ignored.
- Initialization and termination IMODs are not invoked automatically.
- IMODs distributed in the system IMOD library (defined as an ISET called INTERNAL) are available and link edited into the SRVBATCH load module.
- Operator commands are processed if they are entered using the MODIFY (F) or STOP (P) command.
- SRVBATCH terminates when the last active IMOD task completes.
- To set the job step completion code, you can use the BATCHRC() function.

Note: For information about CA-GSS initialization parameters and extended functions, see the *Reference Guide*.

Execute a Compiled IMOD Using SRVBATCH

To execute a compiled IMOD in batch mode, update the following sample JCL procedure to suit your requirements:

Note: All JCL statements supported by the CA-GSS primary subsystem are valid for SRVBATCH.

```

❶//          EXEC PGM=SRVBATCH,PARM='imod_name/arg_string'
❷//PARMLIB  DD DISP=SHR,DSN=CAI.CAW00PTN(RUNPARM)
❸//ISRVLOG  DD SYSOUT=*
❹//IMOD     DD DSN=iset_dsn_1
❺//SYSIN    DD ...
❻ SSNAME ISRV
   ISET iset_name DSN iset_dsn_2 LOAD SSID ISRV
//

```

❶ Replace *imod_name* with the name of the IMOD to be executed, and *arg_string* with a string to be passed both as the initial argument and in the *imod_msg.1* special variable. Omit the slash (/) if no argument is to be passed.

❷ The data set referenced by the PARMLIB DD statement is identical to that in the CA-GSS started task procedure. However, SRVBATCH does not support the INCLUDE initialization parameter. You can omit the data set without error. However, if you include it (for example, because you want to test or specify values), specified values must be correct.

Note: If you use the LOADIMOD() function to fetch additional IMODs, you must include the appropriate ISET initialization parameter statements. Alternatively, you can use the LOAD option of the ISET initialization parameter to load all the IMODs in the ISET.

❸ The data set referenced by the ISRVLOG DD statement is identical to that in the CA-GSS started task procedure. It provides information on IMOD execution, and serves as the target for TRACE and SAY output. You can omit this DD statement if the function is not required.

❹ The IMOD DD statement identifies the ISET that contains the IMOD specified in the PARM parameter. It is required only if the IMOD to be executed is not loaded by an ISET initialization parameter. If it is specified and the IMOD is already loaded, the IMOD is reloaded.

⑤ (Optional) When the stack is depleted, further PULL operations cause SRVBATCH to read data from SYSIN. End of file is signaled by the return of a NULL record. When a NULL record is returned, SYSIN is closed, and a subsequent attempt to read data reopens it. SYSIN can be read multiple times.

You can test for a NULL record using the LENGTH() REXX built-in function or the == comparison operator.

⑥ These last two JCL statements are required for loading IMODs.

Example: IMOD for Processing Operator Commands

This WHOHAS IMOD processes an operator command.

Note: This IMOD is a sample only; it may not work in your environment.

```
#DESC Command: WHOHAS dsn
/*
This IMOD will notify the operator of current owners and waiters
for the data set. Add the next line to the CA-GSS initialization
parameters and cycle the address space.
    COMMAND VERB WHOHAS IMOD WHOHAS
*/
parse upper arg . dsn .
queue 'SRV000 WHOHAS'
queue 'DSNAME = '||dsn
handle = dsnenq('obtain')
if rc ^= 0 then do
    say 'dsnenq.obtain returned '||rc||' - '||handle
    queue '*** Internal error 1 ***'
    queue rc||' - '||handle
    signal write2
end
result = dsnenq('info',handle,dsn)
if rc ^= 0 then do
    if rc ^= 121 then do
        say 'dsnenq.obtain returned '||rc||' - '||handle
        queue '*** Internal error 2 ***'
        queue rc||' - '||result
        signal write
    end
    dsn_count = 0
end
if dsn_count < 1 then do
    queue = '*** Data set not in use ***'
    signal write
end
```

```

/* Tell operator who has data set */
queue 'Jobname System ASID Jobtype Ownership'
queue '-----'
do i = 1 to dsn_count
  queue  dsn_jobname.i||' '|| ,
        substr(dsn_sysid.i,1,8)||' '|| ,
        d2x(dsn_asid.i,4)||' '|| ,
        substr(dsn_jobtype.i,1,8)||' '|| ,
        substr(dsn_status.i,1,4)||' '|| ,
        substr(dsn_mode.i,1,4)
end
write:
  result = dsnenq('release',handle)
write2:
  result = mlwto(,,5,imod_console)
return

```

The IMOD execution process is as follows:

1. An operator enters the WHOHAS command followed by a data set name at an operator console.
2. ISERVE intercepts the command and executes the IMOD.
3. The IMOD obtains the command string as a REXX argument and extracts the data set name.
4. The DSNENQ() function issues a query to GRS about the data set.
5. The IMOD builds a screen display, line by line, on the program stack.
6. The MLWTO() function displays the results on the console of the operator.

The following sample displays the current users of a data set and those users who are waiting for the data set:

```

17:45:40.22          WHOHAS KBROWNE.TEST.CAICLIB
17:45:40.88 STC05434 SRV000 WHOHAS 710
DSNAME = KBROWNE.TEST.CAICLIB
JOBNAME SYSTEM  ASID JOBTYP  OWNERSHIP
-----
KBROWNE MVSBI   00B8 TSU     OWN  SHR
BATJOB  MVSBI   00C8 JOB     WAIT EXCL

```

Package IMODs as a Load Module

If you have a set of IMODs that you use regularly (for instance, a reporting program run in a production environment), you can convert them into a stand-alone, executable load module. This eliminates using SRVBATCH and its PARM and control information.

The load module can be executed like any other load module. There are no required runtime libraries or data sets, except those required by the IMODs themselves. The contents of the EXEC card PARM field are passed to the entry IMOD as the argument string. The contents of the SYSIN data set, if one is provided, can be read by PULL and PARSE PULL REXX instructions in the absence of other stack contents.

To package IMODs as a load module

1. Test the IMODs to ensure that they are working before beginning load module conversion.

The IMODs work correctly.

2. Execute the CAW0LOAD(SRVMAINT) batch maintenance program with the PACKAGE command.

The IMODs are converted to object decks and placed in PDS members.

Note: For information about the PACKAGE batch maintenance command, see the *Reference Guide*.

3. Link edit the packaged IMODs with the CAW0LOAD(SRVBASE) load module.

The executable load module is created.

Example: Create a Reporting Program

You want to create a program called REPORT by packaging all IMODs that are prefixed REPORT_ in an ISET called USER, and the IMODs DATA7 and VERIFY in the ISET called SYSTEM. You want program execution to start with the IMOD REPORT_BEGIN. During execution, the program requires the use of ADDRESS IDCAMS and ADDRESS COMMAND.

```
//STEP1 EXEC PGM=SRVMAINT,REGION=4M
//STEPLIB DD DSN=CAI.CAW0LOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//USER DD DSN=CAI.IMOD.USER,DISP=SHR
//SYSTEM DD DSN=CAI.IMOD.SYSTEM,DISP=SHR
//PDS DD DSN=CAI.PDS.TEMP,DISP=(OLD,PASS)
//SYSIN DD *
```

```

NAME_LIST &LIST1 INCLUDE USER /REPORT_&/
NAME_LIST &LIST2 INCLUDE NAME VERIFY INCLUDE NAME DATA7
PACKAGE IMOD &LIST1 FROM USER      TO PDS
PACKAGE IMOD &LIST2 FROM SYSTEM    TO PDS
PACKAGE PARM ADDRESS IDCAMS IDCAMS 15
PACKAGE PARM ADDRESS COMMAND GSVXAPIE 15 DETACH TYPE 0
PACKAGE COMPLETE ENTRY REPORT_BEGIN TO PDS
/*
//LKED   EXEC PGM=IEWL,PARM='LIST,MAP,RENT',REGION=4M,COND=(0,NE)
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLMOD DD DSN=user.LinkLib,DISP=SHR
//SYSLIB DD DSN=CAI.PDS.TEMP,DISP=OLD
//BASE   DD DSN=CAI.CAW@LOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSLIN DD *
ORDER SRVBASE(P)
INCLUDE SYSLMOD(SRVBASE)
INCLUDE SYSLIB(IMODLIST)
INCLUDE SYSLIB(IMODPARM)
ENTRY SRVBASE
SETCODE AC(1)
NAME REPORT(R)

```

Debugging Considerations

If necessary, you can include both ISRVLOG and PARMLIB DD statements. If present, these data sets function in the same manner as in the CA-GSS started task.

Note: If using the PARMLIB DD statement, it must define a sequential data set or a PDS with a particular member specified (for example, A.B(*member_name*)).

Back Up IMODs

If you write or modify IMODs, you should create backup copies of the ISET data sets that contain them in case these ISETs are damaged by system failure or by accident.

To back up IMODs, use the IDCAMS REPRO function in the following sample JCL procedure:

```
//REPRO      EXEC PGM=IDCAMS
//SYSPRINT   DD SYSOUT=*
//OUTPUT1    DD DSN=CAI.BACKUP.IMOD1,
//           UNIT=TAPE,VOL=SER=BACK01,
//           DCB=(RECFM=VB,LRECL=8000,BLKSIZE=16004),
//           DISP=(,PASS),LABEL=1
//IMOD1      DD DSN=CAI.IMOD1,DISP=SHR
//SYSIN      DD *
             REPRO IFILE(IMOD1) OFILE(OUTPUT1)
//
```

The output is a sequential file that you can store on tape or disk.

Restore an ISET

To restore an ISET, use the IDCAMS REPRO function in the following sample JCL procedure:

```
//REPRO      EXEC PGM=IDCAMS
//SYSPRINT   DD SYSOUT=*
//INPUT1     DD DSN=CAI.BACKUP.IMOD1,
//           UNIT=TAPE,VOL=SER=BACK01,
//           DISP=OLD,LABEL=1
//IMOD1      DD DSN=CAI.IMOD1,DISP=OLD
//SYSIN      DD *
             REPRO IFILE(INPUT1) OFILE(IMOD1) REUSE
//
```

Restore a Single IMOD

To restore a single IMOD

1. Restore the ISET backup that contains the IMOD to a newly allocated ISET.
2. Use the SRVMAINT program and the COPY batch maintenance command to copy the IMOD to the target ISET.

Note: For information about the SRVMAINT program and the COPY batch maintenance command, see the *Reference Guide*.

Migrate ISETs from Previous Releases

If you have ISETs that are not distributed with CA-GSS, they can contain IMODs that were compiled under a different release of CA-GSS. Generally, minor differences in compiler and interpreter versions do not create problems. However, to eliminate the potential for error, you can use the CA-GSS SRVMAINT UPGRADE command to recompile back-level IMODs during initialization. This recompilation is done in memory and is not saved. To save the recompilation, submit the UPGRADE command in a batch job.

To migrate ISETs from previous releases

1. Make a copy of the CAWOJCL(BYSUPGR) member.
2. Update the copy to suit your requirements.
A job is created to recompile selected back-level IMODs.
3. Submit the copy as a job.

The IMODs are recompiled for the current release of CA-GSS and saved.

Batch Maintenance of ISETs and ILOGs

The SRVMAINT program lets you work with ISETs and ILOGs in a batch environment.

The following JCL statements show you how to use SRVMAINT:

```

❶//          EXEC PGM=SRVMAINT[,PARM=LINES=nn]
//SYSPRINT DD SYSOUT=*          /* Diagnostic file (req) */
❷//name1     DD DSN=iset_dsn
❸//name2     DD DSN=data_file_dsn
❹//name3     DD SYSOUT=*
❺//SYSIN     DD *
  (Insert commands here)
/*
//

```

❶ You can use the PARM parameter to specify the number of lines to print per report page and in SYSPRINT. This value can be overridden by commands.

❷ This statement specifies the ISET on which to operate. You can include multiple DD statements to specify multiple ISETs.

❸ This statement specifies a sequential file or PDS to read or write to. You can include multiple DD statements to specify multiple sequential files and PDSs.

④ This statement specifies a report file.

⑤ You insert your batch maintenance commands here.

Note: For information about batch maintenance commands, see the *Reference Guide*.

SRVMANT is implemented using IMODs. If you need debugging information, include an ISRVLOG DD statement.

Important! You must *not* use the following reserved ddnames, except as defined by IBM: JOBLIB, STEPLIB, SYSABEND, SYSIN, SYSMDUMP, SYSPRINT, and SYSUDUMP.

Chapter 10: Common Address Space Shell

This section contains the following topics:

[Overview](#) (see page 335)

[Security](#) (see page 336)

[Create a Common Address Space through a z/OS START command](#) (see page 336)

[Create a Common Address Space as a Batch Job](#) (see page 337)

[Run a Server Application in a Common Address Space Shell](#) (see page 337)

[Customize a Common Address Space Shell](#) (see page 337)

Overview

The purpose of the Common Address Space Shell (CASRV) is to provide an environment where CA components or CA products can be hosted in a secure environment with the proper z/OS integrity while at the same time providing a consistent operational behavior for our customers. The CASRV can handle all console command and response traffic. The CASRV can significantly reduce the amount of time necessary to develop new common services.

This chapter provides general overview information about the CASRV. Specific information on its usage, including examples, for particular CA applications is provided in documentation related to such an application.

You have two options to create a CASRV:

- [z/OS Start Command](#) (see page 336)
- [Batch Job](#) (see page 337)

The CASRV supports operator commands using the z/OS MODIFY/STOP interface. The CASRV also reads commands from an external file and an internal source during initialization. This functionality enables automatic start of CA product or components in the various instances of the CASRV.

Important! For command details, see the *Reference Guide*.

Security

The Common Address Space Shell (CASRV) is intended to execute as a privileged system component. The CASRV control program executes in supervisor state with a system storage protect key of 4 to provide isolation and integrity between the CASRV control program and any hosted CA product or component.

The CASRV control program establishes a trusted security environment. All CASRV control program tasks are associated with this security environment. All tasks of the hosted CA product or component execute with the address space level security environment.

Create a Common Address Space through a z/OS START command

You can create a Common Address Space Shell (CASRV) by using a z/OS START command.

Using this method requires a JCL PROC. The z/OS system-level JCL PROC, IEESYSAS, can be used to start a CASRV. You can also use a component-specific JCL PROC, but it is not required.

The following scenarios let you pass internal commands through the START CIB:

- A CASRV that is created with a START command
- Full-function start address space created with an ASCRE CREATE

A CASRV retrieves any such internal commands from the START CIB during initialization.

A CASRV created with a START command also examines the PARM= field on the EXEC statement image. Any parameters specified on the PARM= field are retrieved during initialization and are processed as internal commands.

Example: z/OS-Supplied JCL Proc

Use the following z/OS START command to create the CASRV using the z/OS-supplied system address space JCL PROC:

```
S IEESYSAS,PROG=CASRVASI,JOBNAME=CASERV,SUB=MSTR,TIME=1440
```

Using this method of starting the CASRV requires that the CASRV component modules reside in a LINKLIST resident library. Furthermore, parameter specifications, if provided, must be defined in the z/OS logical PARMLIB data sets. The SUB=MSTR,TIME=1440 keywords can be eliminated from the START command if you want to run the CASRV under the primary job entry subsystem. However, the CASRV should function in both environments.

Create a Common Address Space as a Batch Job

You can create a Common Address Space Shell (CASRV) and execute it as a batch job, if necessary.

A CASRV that is created as a batch job examines the PARM= field on the EXEC statement image. Any parameters that are specified on the PARM= field are retrieved during initialization and are processed as internal commands.

Run a Server Application in a Common Address Space Shell

An application server is initiated in a CASRV through the use of an ATTACH command. The CASRV supports hosting multiple server applications in a single address space. An ATTACH command initiates each server, respectively. Each server application executes in a unique job-task task tree environment, separate from other server application servers and isolated from the control program job-step task tree.

Important! For command details, see the *Reference Guide*.

Customize a Common Address Space Shell

Because a Common Address Space Shell (CASRV) must support execution in early IPL environments, no explicit dependency exists in the following areas:

- JCL PROC
- STEPLIB
- Externally allocated parameter data set.

The CASRV uses the CA Common Service PARMLIB reader service; therefore, command statements can be obtained through the z/OS system logical PARMLIB data sets, or through an externally allocated data set. Furthermore, internal commands can be passed when the address space is created as previously described.

The CASRV control program commands provide the basis for customizing its environment as well as the server application that executes within it. The CASRV does not support any explicit initialization statements, but the commands passed as internal commands to an address space create, and any commands read from an external data set, effectively function as initialization statements.

Many of the CASRV control program commands serve to customize the address space for a server application. To some degree, the CASRV functions as a z/OS INITIATOR because it lets you allocate resources for a hosted server application before attaching the server application. By including the appropriate CASRV control program commands in an internal source or in an external data set or PARMLIB member, data sets can be allocated, message tables can be established, and then server applications can be started in a CASRV.

Appendix A: Troubleshooting

This section contains information about:

- Diagnosing and resolving problems.
- Receiving ongoing product releases and maintenance.
- Requesting product enhancements.

We recommend that you follow CA Technologies product specific diagnostic options before applying the guidelines in this chapter.

For online technical assistance and a complete list of locations, primary service hours, and telephone numbers, contact CA Support at <http://ca.com/support>.

This section contains the following topics:

- [Collect Diagnostic Data](#) (see page 339)
- [Interpret Diagnostic Data](#) (see page 341)
- [Identify and Resolve the Problem](#) (see page 341)
- [CA4FIVP Options](#) (see page 353)
- [CA TLC: Total License Care](#) (see page 353)
- [Product Versions and Maintenance](#) (see page 354)
- [Request Enhancements](#) (see page 354)

Collect Diagnostic Data

If you encounter a CA Technologies product or operating system/subsystem error you suspect is caused by a CA Common Services for z/OS component, collect the appropriate information from the following list of available diagnostics:

Service	Diagnostics
CAIRIM	<ul style="list-style-type: none">■ CAISMFU and CAIRIMU utilities output■ Console messages■ SVC dumps

Service	Diagnostics
CAIENF and CAIENF/CICS	<ul style="list-style-type: none"> ■ Console messages ■ CAIENF operator commands output ■ The output from either the 'ENF SVCDUMP' command or the 'ENF DUMP' command ■ Traces ■ SVC DUMP of any problem CICS region
CAICCI	<ul style="list-style-type: none"> ■ Console messages ■ CAICCI operator commands output ■ SVC dumps ■ Traces
Datacom/AD	<ul style="list-style-type: none"> ■ CA4FIVP
Event Management	<ul style="list-style-type: none"> ■ syslogd messages ■ Trace data in CAIGLBL0000/emsvrc/config/debug.log (if the CA_CAIDEBUG, CAIOPR_DEBUG, and/or CAISTARDEBUG environment variables are set and the traceinfo file exists in that same directory) ■ Web Server error log ■ oplog messages for the day ■ CEEDUMP ■ java trace if using JAVA GUI
Agent Technology	<ul style="list-style-type: none"> ■ Contents of the STDOUT and STDERR for the batch jobs that start Agent Technology and the individual Agents ■ Trace data in the aws_orb.log, aws_sadmin.log and awsservices.log files found in the directory \$AGENTWORKS_DIR/services/var/log ■ Contents of the ENVFILE used by Agent Technology and the individual agents (These should be identical; differences can cause problems.) ■ Output of the TSO netstat command or the onetstat command showing port usage ■ Output of the batch job AWFTEST

Interpret Diagnostic Data

When you have collected the specified diagnostic data, write down your answers to the following questions:

1. What was the sequence of events prior to the error condition?
2. What circumstances existed when the problem occurred and what action did you take?
3. Has this situation occurred before? What was different then?
4. Did the problem occur after a particular PTF was applied or after a new release of the software was installed?
5. Have you recently installed a new release of the operating system?
6. Has the hardware configuration (tape drives, disk drives, and so forth) changed?

From your response to these questions and the diagnostic data, try to identify the cause and resolve the problem.

Identify and Resolve the Problem

From your response to these questions and the diagnostic data, try to identify the cause and resolve the problem. You may find the following checklists useful:

- [CAIENF/CICS Checklist](#) (see page 342)
- [CAIENF/DB2 Checklist](#) (see page 343)
- [CAIENF/USS Checklist](#) (see page 344)
- [CAICCI Checklist](#) (see page 345)
- [Event Management Checklist](#) (see page 348)
- [Agent Technology Checklist](#) (see page 351)

CAIENF/CICS Troubleshooting Checklist

The following items may help you identify the problem.

1. Is the proper CAIENF/CICS intercept module present in the CAW0LOAD data set?

Check the CAS9Cxx Load Module. The value for xx should match the version of CICS/TS you are running. Valid values are as follows:

CAS9Cxx Value	Version of CICS/TS
64	3.1.0
65	3.2.0
66	4.1.0

2. Is the CAW0LOAD in the z/OS LINKLIST or STEPLIB concatenation?

Add it if it is not.

3. Are all of the required DCMs present in the CAIENF startup procedure?

If a required DCM is missing, add the DCM configuration statement to the CAIENF ENFPARMS data set. Some frequently-used DCMs are: CAS9DCM2, KO50DCM2, and J161DCM2.

4. Is the CA Technologies application initialization module available to CAIENF through the STEPLIB or the LINKLIST?

If it is not, make it available. Some frequently used application initialization modules are: TSSTCINT, ACFAEINT, and CAKSCINT.

5. Check how and where the intercept module is being loaded.

- Is it loaded into the CICS private using the CICS STEPLIB or the CENFLIB DD statement?
- Is it loaded dynamically into the MVS CSA using the CICSREL parameter?
- Is it automatically activated using the MODE(CICS,ON) parameter?
- Is it manually activated with CICS(START,jobname,product)? This method is not supported for CA ACF2 and CA Top Secret.

6. Is the module CAILPAM located in the CAI.CAWOLINK data set?

If CAILPAM is not in the system linklist, and the intercept module is being loaded in CICS private, then CAILPAM must be found in the CICS STEPLIB.

7. Is the problem happening in all CICS regions?
If **yes**:
 - Are all regions running the same release?
 - What release of MVS and JES are running?
 - Did it ever work? If it did, what changed?If **no**:
 - What regions does it work in?
 - Check how the intercept module is being loaded.
8. Are you getting any messages (like TSS/CICS Phase 1 Initialization... or ACFAE023 CA ACF2 CICS Waiting on CAIENF/CICS Initialization.)?
 - DUMPT module=CASESSI csect=CASEWTO. You should see an eyecatcher for DFH1500, DFH1517, DFHSI1517, and DFHPA1108.
 - DUMPT module=CASESSI csect=CASECICS. You should see an eyecatcher for DFH1517 and DFHSI1517.
9. What does the ENF STATUS display indicate?
Is the CICS job name found in the list? Is the APPLID displayed? Is the auto install feature active?

CAIENF/DB2 Troubleshooting Checklist

The following items may help you identify the problem.

1. Is the CAW0LOAD in the MVS LINKLIST or STEPLIB concatenation?
Add it if it is not.
2. Is the proper CAIENF/DB2 intercept module present in CAW0LOAD?
The load module is CASR230.
3. Is the CA Technologies application initialization module (CADB2INT) available to CAIENF through the STEPLIB or LINKLIST?
If it is not, make it available.
4. Are the CAS9800I and CAS9801I initialization messages issued?

5. Are all required DCMs present (installed) in the CAIENF startup procedure?

If a required DCM is missing, install it into the database. Some frequently used DCMs are: CAS9DCM0 and DB10DCM1. Note that the DCM supplied by the CA security products is the same one.

6. Is the intercept module being loaded?

- Check for the DB2REL statement in ENFPARMS.
- Check MVS storage. Is it "Go to address EVT+430???"? The ??? means there are three levels of indirect addressing. It should point to the vector table, and the first CSECT should be at +X'70'.

Note: You can get the EVT address by issuing the ENF MAP command.

CAIENF/USS Troubleshooting Checklist

The following items may help you identify the problem.

1. Are z/OS UNIX Services initialized properly and running in full function mode?

CAIENF/USS does not install any of its intercepts until the UNIX kernel successfully initializes, so a z/OS USS configuration problem will prevent CAIENF/USS from starting.

Without CAIENF/USS running, verify that you can initiate an interactive UNIX shell session using OMVS or a Telnet session.

2. Are there any unusual startup messages from CAIENF?

CAIENF/USS initializes during CAIENF startup, and many times, unusual conditions are detected and reported at this point.

3. Is the DCM module for CAIENF/USS installed, along with the DCMs for any products that require CAIENF/USS?

4. Are your security settings correct?

The user that the CAIENF address space runs with must be defined with z/OS UNIX Services superuser privileges (UID 0).

5. In the event of an extreme error, CAIENF/USS will generate a system SVC dump. These dumps should be retained for CA service personnel.

If you suspect a problem with CAIENF/USS and are trying to gather your own dumps, you need to ensure that you include the OMVS address space along with any applications you suspect are having problems.

Also, CAIENF/USS uses a dataspace named "#UVT#", and this dataspace should also be included in any system dumps you generate. Due to the amount of data involved, you may need to create larger dump data sets.

CAICCI Troubleshooting Checklist

The following items may help you identify the problem.

1. Is the CAICCI subsystem operational?

For example, if you are receiving message CAS9626I, check to see if the CAICCI subsystem is operational.

2. Have the CAICCI parameters been updated? Are they being referenced by the CAIENF procedure?

See the chapter [Common Communications Interface](#) (see page 163) in this guide for instructions and examples.

3. Obtain the necessary traces to use in debugging.

CCIPC TRACE

Enable trace box from CAICCI/PC Configurator Properties dialog. You can also enable additional SSL tracing at the direction of CA Support by checking the DumpSSL box. The trace output goes to the directory specified on the Trace tab of the Configurator Properties dialog.

CAICCI MVS LOCAL TRACE

The following z/OS commands are issued from the console:

```
F ENF,CCI(LWTL) * Activate trace
F ENF,CCI(NLWTL) * Deactivate trace
```

CAICCI MVS REMOTE TRACE

This z/OS command is issued from the console immediately after the error occurs:

```
F ENF,CCI(PRINTT)
```

The trace will be written to a TRCPRINT DD that will be dynamically allocated by CAIENF.

Note: If CAIENF is running as a subsystem, it is necessary to specify a TRCPRINT DD statement in the CAIENF procedure with LRECL=133.

You cannot specify a SYSOUT data set in this case, and if you specify a catalogued data set, it must be catalogued in the master catalog.

CAICCI MVS TCP/IP PROTOCOL TRACES

The following z/OS commands should be issued from the z/OS console, substituting the name of the started task (CCITCP, CCISSL, CCITCPGW or CCISSLGW) for NNNNNN:

```
F NNNNNN,T,SYSPRINT * Activate the trace
F NNNNNN,NOTRACE * Deactivate the trace
```

Note: Be sure that a SYSPRINT DD is specified in the proc for the started task. Shut down the started task afterwards to obtain all messages. Request the entire job output.

CAICCI MVS COMPONENT TRACE

The following z/OS command, issued from the z/OS console, will activate CAICCI Component Tracing:

```
TRACE CT, ,COMP=CACCI,PARM=CTECCI00
```

CTECCI00 is shipped in CAI.CAW0OPTN and must be moved to an MVS PARMLIB data set. As provided in the Option Library, the CTECCI00 member starts tracing and forwarding output to an External Writer. A sample PROC, CCIXWTR, is provided in CAI.CAW0PROC, which invokes the IBM external writer ITTRCWR. This proc must be placed into SYS1.PROCLIB and be customized to define the external trace data set(s).

The External Writer copies the Component Trace output into the external trace data sets. The trace records within the data sets can later be processed and examined using a specialized CAICCI formatter running under IPCS.

An SVC dump of Data Space #CCICT# which belongs to the CAMASTER address space (DSPNAME=CAMASTER.#CCICT#) can also obtain the Component Trace records.

Currently CAICCI Component Trace only traces local CCI processing.

USS Environment Troubleshooting

Due to the nature of running in the USS environment, the debugging of problems can be quite different than on the mainframe.

Check Job Output

The EXEC program BPXBATCH uses STDOUT and STDERR for output. These files are kept on the HFS. The script or program that is executed can be on the PARM statement or in the STDIN DD statement.

A CAWOJCL member NSMEMCHK has been provided to check some of the main sources of problems. Run the job and then verify that the output is correct.

The job checks things like the user ID that is being used to start the job, the amount of free space in /tmp, the results of certain TCP/IP functions, and so on.

Check Environment Variable Settings

A main cause of problems is the setting of environment variables.

To determine if the settings are in place and correct:

- To display current environment variable settings, issue the set command.
- If the problem is seen when running a script, adding a 'set -x' command in the script displays the settings in STDOUT when the problem arises.

Check Space on the zFS File

Another common problem is running out of space on the zFS file. Even though the file is physically a linear VSAM data set, it can run out of space, especially if all log files are retained. To determine how much space is left on a zFS, issue the following OMVS command to display the current amount of space left on all the file systems currently mounted:

```
df -k
```

Check Log Files

If Event Management has a problem writing to its log files, it writes a message to syslogd. Check your syslogd log file for any messages like this:

```
EDC5133I No space left on device
```

To direct these messages to the system console, add the following entry to your syslogd.conf file:

```
*.info /dev/console
```

Check Security Permissions

Inadequate security permissions can also cause problems during the installation process or during execution of the started tasks. See the product installation and maintenance documentation for security permissions required for the user ID used to install the product and for the user IDs assigned to the started tasks.

Event Management Troubleshooting Checklist

The following items may help you identify the problem.

1. Are z/OS UNIX Services configured and operational?

You should be able to initiate a USS shell session and issue various UNIX commands to verify that z/OS UNIX Services are operating correctly. Pay particular attention to the USS configuration parameters specified in the *Installation Guide*.

2. Are proper security definitions in place?

Event Management components must run with superuser privileges (UID 0). Verify that caiopr and stardaemon both start and run with UID 0. If you have activated BPX.DAEMON support, the user ID running Event Management needs to be permitted to the BPX.DAEMON and BPX.SERVER FACILITY resources.

In addition, depending on the z/OS release you have installed, this ID may also require both SURROGAT and BPX.SERVER.userid permissions for any users that run Event Management commands.

3. Is a compatible Java environment installed?

The Event Management Java GUI requires Java at a JDK 1.1.6, 1.1.8, 1.3.1 or 1.4 level (available from IBM). Verify that simple Java programs (such as the JDK supplied samples) compile and execute properly.

4. Is your web server configured and running properly?

You should be able to request a simple HTML document by connecting a web browser to the mainframe and requesting one of the CA Common Services for z/OS HTML documents. If, for example, your host name is "mainframes.com" and you have selected port 4080 to run the Services web server, then this URL should display the CA Common Services for z/OS welcome page:

`http://mainframes.com:4080/tngfw/tngfw.html`

5. Are all of the required processes running?

From a UNIX shell session or batch job with superuser privileges, issue "ps -ef" (or the z/OS command "D OMVS,A=ALL") and verify that these processes are running:

- Event Management opr component requires caiopr, cailgr and newdaylog processes. Optionally, if store and forward is required, an oprsafd processing should be running.
- Star component requires stardaemon process
- Calendar component requires calendar process
- Java running class CA.Unicenter.comm.w2RmiServerImp
- logonsrvr.exe
- w2Tree.exe
- CaemRtS
- EMServer.exe

6. Are your environment variable settings correct?

There are a number of environment variables that must be correctly set in order for Event Management to operate correctly. These environment variables are set in the \$CAIGLBL0000/PROFILE file for the Event Management processes and in the \$CAIGLBL0000/browser/httpd.envvars file for the web server and Java GUI processes.

Tracing is usually turned on by setting environment variables before the process is started. The output is normally directed to STDOUT. The following variables are used to start tracing for caiopr. Additional environment variables for tracing other specific processing can be set at the direction of CA Support.

```
CA_CAIDEBUG=Y  
CAI_NODENAME_DEBUG=Y
```

Some trace messages will be written based on the existence of file /cai/nsmem/emsvc/config/traceinfo. That file contains control statements that direct the trace to a file, syslogd, or both. Send them to a file because it is easier to send the file for analysis.

The trace file that is written to is /cai/nsmem/emsvc/config/debug.log.

Note: For more information, see this guide and the *Reference Guide*.

7. Are the CA Datacom/AD libraries in your STEPLIB concatenation?

z/OS UNIX Services processes require the CA Datacom/AD CUSLIB and CAAXLOAD data sets to be defined in the STEPLIB environment variable.

8. If you are having trouble connecting Event Management to other platforms, you may need to configure CAICCI on z/OS and on the remote system. For more information on defining remote nodes to the z/OS system, see the Common Communications Interface chapter.

9. For more information about defining the z/OS system to the remote distributed machine, see the CA NSM documentation.
10. If you are having problems logging on to the Java GUI from your web browser, verify that the program `logonserver.exe` in directory `$CAIGLBL0000/ww/bin` is running APF authorized and program controlled.

You can verify this with the following USS command:

```
extattr $CAIGLBL0000/ww/bin/Logonserver.exe
```

11. If you are having trouble capturing syslogd messages, verify that the syslogd daemon is running, and that the syslogd configuration file correctly forwards messages to Event Management. This can be checked by ensuring that the `/etc/syslog.conf` file contains a line for the node's pipe file such as:

```
.info.../cai/nsmem/opr/config/USCMCT30/pipe/oprpipe0001
```

12. If you are having trouble running CATRAPD, make sure that SNMP port 162 is available and can be opened by CATRAPD. Your TCP/IP configuration file can prevent certain ports (such as 162) from being opened except by the specified application.

By default, IBM ships TCP/IP pre configured so that port 162 can be opened only by a program called `SNMPQE`, and this must be removed before CATRAPD can operate correctly. Alternately, you can specify a different port by editing and exporting an environment variable. For more information, see the *Installation Guide*.

Agent Technology Troubleshooting Checklist

The following items may help you identify the problem.

1. Are z/OS UNIX Services configured and operational?

You should be able to initiate a USS shell session and issue various UNIX commands to verify that z/OS UNIX Services itself is operating correctly. Pay particular attention to the USS configuration parameters specified in the *Installation Guide*.

2. Are proper security definitions in place?

The user that starts the services must belong to the same group as the owning group for the Agent Technology files. The userid should NOT be `UID(0)`.

3. Are all the required processes running?

Full Agent Technology requires `awservices`, `aws_orb` and `aws_sadmin` processes to be running. From a UNIX shell session or batch job with superuser (`su`) privileges, issue `"ps - ef"` (or the z/OS command `"D OMVS,A=ALL"`) and verify that these processes are running.

4. Are your environment variable settings correct?

You must correctly set a number of environment variables for Agent Technology to operate correctly. Set these environment variables in the \$AGENTWORKS_DIR/agentworks.profile file for the Agent Technology processes. A subset of these variables must be available to the Agent Technology started task and to any task that starts an individual Agent. This subset of environment variable settings is located in the file referenced by the ENVFILE DD in the Agent Technology started task JCL. A sample of the ENVFILE is delivered in the Common Services CAW0OPTV library. The ENVFILE data set must have the same DCB attributes as the CAW0OPTV library.

Note: For more information, see this guide and the *Reference Guide*.

5. If you are unable to see a mainframe Agent at the manager machine, ensure that the mainframe is classified as an IBM3090 machine. Additionally ensure that the \$AGENTWORKS_DIR/services/config/aws_sadmin.cfg file contains the correct SNMP_COMMUNITY and SNMP Trap destination information to direct the traps to the manager machine. Also use the awsservices list command to ensure that the Agent is actually running.

6. If both Agent Technology and the individual Agent are running but are not communicating, check if your system is running more than one TCP/IP stack. If so, you should add a SYSTCPD DD to your Agent Technology startup proc and to the Agent's startup proc. Additionally, both agentworks.profile and ENVFILE should contain the following 2 environment variables:

```
_BPXK_SETIBMOPT_TRANSPORT=NNNNNN  
RESOLVER_CONFIG="//'VTAM.TCPIP.TCPIP.DATA' "
```

where NNNNNN specifies the name of the TCP/IP task to connect to, and 'VTAM.TCPIP.TCPIP.DATA' points to the SYSTCPD for the TCP/IP task.

CA4FIVP Options

You can specify the following CA4FIVP options:

V

Verbose option. Prints all validation messages and up to 100 lines of each table, template, and profile member.

Axx

Validates the control tables for ViewPoint and the control tables for CA Technologies product/ViewPoint. xx is the two digit product code of the base product.

Lnnn

Prints up to nnn lines from each table, template, and profile member.

LO

Prints all lines comprising each table, template, and profile member. This option is only valid when used with V.

As illustrated in the following examples, you can combine options.

- To validate ViewPoint and all installed ViewPoint product options, specify:
CA4FIVP
- To validate ViewPoint and the CA-product/ViewPoint that is identified by the 2 digit product code YY:
CA4FIVP AYY
- To process like example 2 plus print up to 100 lines of each table, profile, and template member, specify:
CA4FIVP V AYY
- To process like example 1 plus print up to 200 lines from each ViewPoint table, profile, and template member as well as the control tables for each installed ViewPoint product option, specify:
CA4FIVP V L200

CA TLC: Total License Care

Many CA Technologies software solutions use license keys or authorization codes to validate your hardware configuration. If you need assistance obtaining a license key or authorization code, contact the CA-TLC: Total License Care group through <http://ca.com/support>.

Product Versions and Maintenance

CA Common Services for z/OS is packaged together with a CA Technologies solution. The CA Technologies solution that uses the various services includes requirements for specific versions and levels of CA Common Services for z/OS. In all cases, you should consult the CA solution documentation for specific component usage details and installation recommendations.

As new features are incorporated into CA Technologies solutions, CA Common Services for z/OS is reissued to include the new functionality. If you already have some CA Common Services for z/OS components installed, you only need to install any additional services or features required by the CA Technologies product. You may reinstall a new genlevel of CA Common Services for z/OS at any time to take advantage of new features or higher service levels.

CA Common Services for z/OS maintenance updates containing service upgrades, problem resolutions, or both will also be distributed to all users with current maintenance agreements. This maintenance update, typically shipped four times yearly, allows you to keep your components up to date and to avoid encountering known problems. Clients are encouraged to apply the preventive maintenance service in a timely fashion in order to help maintain a trouble free, stable environment. Documentation updates are also performed regularly and provided automatically to current users.

At times, a product may issue both a CA Common Services for z/OS genlevel upgrade and a CA Common Services for z/OS maintenance update, in which case you may need to install new features from the distribution update as well as apply maintenance from the maintenance update. Again, we urge you to consult the product specific documentation for product specific requirements.

Request Enhancements

CA Technologies welcomes your suggestions for product enhancements. All suggestions are considered and acknowledged. You can use either of two methods to request enhancements:

- Enter your request through <http://ca.com/support>, the CA Technologies web-based, interactive support system.
- Contact your account manager or a CA Support representative.

Appendix B: CAIENF Batch Database Query and Administration

This appendix describes commonly used batch database query and administration requests. These requests display information about CAIENF events and allow administration of database event tables. Administrators may design their own database queries.

Important! Updating or deleting tables must only be performed under the advice of CA CAIENF support personnel. CASQL004 and CADB001 must only be used under advice of CA CAIENF support personnel or as instructed by a maintenance PTF.

The following is a list of common requests:

- **CASQL001** - Lists all CAIENF Event database tables.
- **CASQL002** - Lists Event table event instances. A date range may be specified
- **CASQL003** - Lists Event table count of event instances. A date range may be specified.
- **CASQL004** - Deletes (DROP) an CAIENF database event table.
Important! Use this job under advice of CA CAIENF CA Support only.
- **CADB001** - Flushes database cache and closes all CAIENF tables. This job may be used with job CASQL004.

Important! Use this job under advice of CA CAIENF CA Support only.

Index

A

- abends • 231
- ACEE, CA-GSS • 289
- activating store and forward • 52
- ACTIVE • 179
- ADDFILE • 240, 244, 259
- ADDRESS environments, CA products • 299
- Agent Technology
 - agent operation • 76
 - agent protocol • 76
 - agent status • 77
 - architecture • 77
 - Distributed Services Bus • 78
 - Service Control Manager • 77
 - SNMP Administrator • 79
 - aws_admin, store table cache sizes • 89
 - cache sizes for store • 89
 - configuring • 89
 - description • 22
 - Event Management integration • 91
 - managed objects • 76
 - manager/agent concept • 20
 - overview • 75
 - sizing for performance • 89
- agents
 - manager/agent concept • 20
- ARCHIVE • 257
- archive files
 - automatic archive • 258
 - generic JCL • 259
- auto commands • 139
- AUTOCMDS DD • 95
- automated message processing • 25
- automatic archive • 258
- AUTOPRINTT • 179

B

- Berkeley syslog daemon • 38
- BPX BATCH • 29
- buffer pools • 241
- buffers • 238, 241
- BUILD • 179

C

- CA LMP
 - CAIRIMU output • 104
 - console messages • 116
 - Emergency Key Generator (EKG) • 117
 - errors • 121
 - execution key • 110
 - loading new keys • 114
 - operation • 108
 - overview • 108
- CA4FIVP • 353
- CA-GSS
 - ACEE • 289
 - commands
 - ADDRESS IDCAMS • 299
 - PACKAGE • 330
 - DD statements, ISRVLOG • 333
 - debugging
 - IMOD load modules • 331
 - executing
 - compiled IMODs • 327
 - loaded IMODs • 324
 - packaged IMODs • 330
 - initialization parameters
 - GLOBVAL • 314
 - ILOG • 310
 - ISSET • 321
 - ISETs
 - accessing • 321
 - allocating files • 321
 - overview • 276
 - REXX
 - implementation • 296
 - language • 296
 - SRVBASE load module • 330
 - SRVMAINT utility • 330
 - user IDs • 289
 - utilities
 - SRVBATCH • 326
 - SRVMAINT • 310, 321
- CA-GSS, ADDRESS environments
 - CA Jobtrac Job Management • 300
 - CA MIM TPCF • 301
 - CASYSVIEW • 302

-
- CAVIEW • 302
 - ISERVE • 299
 - REXX • 301
 - CA-GSS, functions • 306
 - \$SERVER() • 285, 288
 - SPAWN() • 304
 - STACKINF() • 304, 306
 - CA-GSS, ILOGs
 - accessing • 310
 - allocating files • 310
 - logging WTO messages • 311
 - retrieving records • 311
 - writing records • 311
 - CA-GSS, IMODs
 - coding ADDRESS environments • 318
 - coding functions • 318
 - compiler directives • 315
 - compiling • 323
 - data integrity • 295
 - extended error codes • 303
 - external subroutines • 314
 - ILOG files • 309
 - loading • 323
 - packaging as load modules • 330
 - preparing for execution • 323
 - program stack • 304
 - recovery • 295
 - testing from the EXECUTE panel • 324
 - variable pools • 312
 - variables • 312
 - writing • 322
 - CAICCI
 - Assured Delivery • 195
 - communications protocols • 167
 - Configurator • 204
 - specifying TCP/IP protocol parameters • 205
 - configuring • 188
 - configuring CAICCI SPAWN • 194
 - connections • 200
 - crosssystem coupling facility (XCF) • 191
 - crosssystem extended services (XES) • 192
 - defining
 - LOGGER database • 195
 - non-network configuration • 170
 - SNA network mainframe configuration • 186
 - TCP/IP configuration • 173
 - VTAM resources • 187
 - establishing TCP/IP connections • 170
 - generic resources • 193
 - LOGGER database commands • 197
 - multiple network transports • 203
 - overview • 163
 - parallel sysplex configurations • 190
 - PC implementation • 169
 - planning configuration • 165
 - routing control information • 168
 - spawning service • 163
 - TCP/IP gateway commands • 178
 - troubleshooting • 345
 - CAICCI LOGGER database commands
 - DBSTATUS • 198
 - DISPLAY • 198
 - PURGE • 198
 - RELEASE • 199
 - REPORT • 199
 - STATUS • 199
 - CAICCI SPAWN, configuring • 194
 - CAICCI TCP/IP gateway commands
 - CONNECT • 180
 - DISCONNECT • 180
 - PING • 182
 - RELEASE • 183
 - CAICCI, Windows
 - configuring • 204
 - specifying protocol parameters • 205
 - testing protocol configuration • 210
 - tracing communications problems • 210
 - CAICCI-PC, installing • 204
 - CAIENF
 - architecture • 138
 - auto commands file • 141
 - defining as z/OS subsystem • 139
 - execution
 - started task • 139
 - z/OS auto commands • 139
 - z/OS subsystem • 139
 - master catalog, in • 139
 - option file, sharing • 146
 - overview • 137
 - restarting • 140
 - SNMP Monitor
 - overview • 158
 - starting • 159
 - start options • 141
 - started task JCL member • 139
 - stopping • 140
 - viewing event names • 139
 - CAIENF/CICS
-

- activating CA products • 153
- installing intercepts • 150
 - automatic • 151
 - manual • 153
 - specific CICS regions, in • 152
 - specific CICS releases, for • 152
- intercept activation • 150
- locating modules • 151
- operation • 150
- overview • 150
- CAIENF/CICS SPAWN
 - installing intercepts • 154
 - automatic • 155
 - manual • 155
 - intercept activation • 154
 - locating modules • 154
 - operation • 154
 - overview • 154
- CAIENF/DB2
 - activating CA products • 156
 - installing intercepts • 155, 156
 - intercept activation • 155
 - operation • 155
 - overview • 155
- CAIENF/USS
 - operation • 157
 - overview • 157
 - reinitialization • 157
- CAIRIM
 - auto commands file • 98
 - CAS9INIT program • 130
 - error handling • 99
 - execution • 96
 - automatic • 98
 - batch job • 97
 - started task • 96
 - SUB=MSTR parameter • 97
 - without subsystem support • 96
 - features • 93
 - issuing auto commands • 95
 - operation • 94
 - overview • 93
 - specifying data sets • 95
 - verifying initialization • 95
- CAIRIM utilities
 - CAIRIMU • 104
 - CA LMP problems • 104
 - PROD • 105
 - CAISMFU • 106
 - CAISUBU • 107
 - operation • 104
- CAISSF
 - overview • 125
 - system authorization facility support • 120
 - using CAIRIM to initialize • 130
- CAISUBU • 107
- calendars • 22
 - sharing • 34
 - time controls • 34
- CA-L-Serv • 213
 - abends • 231
 - client applications • 218
 - command members • 221, 222, 223, 224
 - command processing • 215, 224, 225, 231, 232
 - communications server • 215
 - components • 214
 - configuration • 217, 223
 - data buffers • 238, 241, 244
 - errors • 231
 - file groups • 246, 247
 - file server • 233, 249
 - kernel server relationship • 214
 - LDMAMS utility • 252
 - logging • 215, 226, 227, 228, 230
 - LSR buffer pools • 243
 - managed files • 239, 249, 251, 253
 - message tables • 219, 220
 - messaging • 215, 219, 220
 - operating values • 218
 - operation • 214
 - performance • 245
 - relational tables • 222
 - REPRO • 255
 - servers • 216, 221, 225
 - SQL Server • 216
 - starting and stopping • 217
 - system and subsystem names • 217
 - VTAM Communication • 267
 - XCF Communications server • 263
- CAS9INIT program • 130
- cawto command • 27, 28, 29, 31
 - script for calling cawto • 30
- commands
 - CAGSS • 296
 - CAICCI LOGGER database • 197
 - CAICCI TCP/IP gateway • 178
 - Event Management • 35
- communications protocols

- cross memory services • 167
- parallel sysplex • 167
- SNA • 167
- TCP/IP • 167
- communications server • 260
 - displaying information about • 263
 - protocols • 260, 261
 - set up • 262
 - starting and stopping • 262
- components • 21
- COMPRESS • 257
- CONNECT • 180
- console messages reduction, CA LMP • 116

D

- daemons
 - Berkeley syslog • 38
 - catrapd • 53
- data buffers • 238, 239
- date
 - controls, calendar • 22, 34
- DCM statements, configuring • 142
- DISCONNECT • 180
- DISPLAY • 232
- DUMP • 180

E

- Emergency Key Generator (EKG) • 117
- ENF/CICS, troubleshooting • 339, 342
- ENF/DB2, troubleshooting • 339, 343
- ENF/USS, troubleshooting • 344
- ENQ requests • 248
- enqueue names • 295
- environment variables
 - SNMP Monitor • 160
- errors • 231
- Event Console
 - command history • 51
 - configuration buttons • 51
 - customizing • 51
 - held messages • 50
 - log file description • 50
 - log messages • 50
 - record selection • 51
 - viewing multiple log files • 52
- Event Management
 - BPXBATCH • 29
 - commands

- catrap • 54
- cawto • 29, 35
- cawtor • 35
- opreload • 49
- console • 49
- date and time controls • 22
- description • 21
- environment variables • 46
- GUI • 49
- held messages • 50
- log messages • 50
- messaging examples • 28
- multiple console log support • 52
- policies • 49
- routing console messages • 70
- security enhancements • 58
- syslog daemon • 38
- troubleshooting • 348
- WTOR messages • 50
- z/OS interface • 62
- Event Management tasks
 - assigning actions to be performed • 37
 - correlating messages • 48
 - customizing your Event Console • 51
 - defining message action policies • 37
 - distributing message activity • 37
 - establishing date and time controls • 34
 - identifying messages requiring special handling • 36
 - monitoring message traffic • 49
 - putting policy into effect • 49
 - submitting process as another user • 58
 - trapping messages • 34
 - using SNMP • 52
 - view multiple remote console logs • 52
- Event Management Utilities
 - components • 62
 - criteria statements • 66
 - description • 62
 - destination statements • 65
 - drivers • 68
 - transports • 63

F

- file groups • 246, 247
 - characteristics • 246
 - define • 247
 - select • 247

File Server (CA-L-Serv) • 249
 command-members commands • 233
 communications • 236
 configure • 233, 235, 236, 238
 console-issued commands • 234
 data buffers • 238, 239
 information about, displaying • 249
 manage • 249
 multiple-system environment • 236
 outages • 250
 servers, host and remote • 235
 starting and stopping • 233

Framework Event Management Server • 62, 63
 configuration • 69
 criteria statements • 66
 destination statements • 65

H

HELP • 181

I

ILOGs

 accessing • 310
 allocating files • 310
 logging WTO messages • 311
 retrieving records • 311
 SRVMAINT program • 333
 writing records • 311

IMODs

 accessing the editor • 322
 backing up • 332
 CA products, in • 291
 CA-GSS releases, migrating from previous • 333
 coding ADDRESS environments • 318
 coding functions • 318
 compiler directives • 315
 compiling • 323
 data integrity • 295
 debugging load modules • 331
 editor • 293
 executing • 330
 external data queue (EDQ) • 304
 external subroutines • 314
 ILOG files • 309
 loading • 323
 naming conventions • 292
 packaging as load modules • 330
 preparing for execution • 323

 prerequisites • 322
 primary ISERVE • 291
 program stack • 304
 recovery • 295
 secondary ISERVE • 291
 SRVMAINT program • 333
 testing from the editor • 324
 use of extended error codes • 303
 use of stacks • 304
 variable pools • 312
 variables • 312
 writing • 292, 322

INACTIVE • 181

installation of Agent Technology
 sizing for performance • 89

International Standards Organization (ISO) • 55

Internet Assigned Numbers Authority (IANA) • 55

ISETS

 accessing • 321
 allocating files • 321
 CA-GSS releases, migrating from previous • 333
 RACF-secured • 292

K

KILL • 181

L

LOCAL statement, Windows • 177

LOGGER database, defining • 195

LSR buffer pools • 243

M

managed files

 backing up • 254, 255
 compressed file • 256
 deleting content of • 256
 file availability, setting • 252
 LDMAMS utility • 252
 maintain • 251
 REPRO, changing default values for • 255
 restoring • 254

managed objects • 76

management information base (MIB) • 55

manager/agent concept • 20

master catalog • 139

message action

 distributing activity • 37
 keywords • 38

- policy • 37
- restriction • 44
- servers • 37
- message action keywords
 - ALLOW • 38
 - AUTORPLY • 38
 - BANNER • 38
 - COMMAND • 38
 - DELAY • 38
 - DELKEEP • 38
 - DISABLE • 38
 - DISCARD • 38
 - ENABLE • 38
 - EVALUATE • 38
 - EXIT • 38
 - EXPORT • 38
 - FORWARD • 38
 - GOTO • 38
 - HILITE • 38
 - IGNORE • 38
 - PERMIT • 38
 - PREVENT • 38
 - PROHIBIT • 38
 - SENDKEEP • 38
 - SENDOPER • 38
 - SENDUSER • 38
 - SUPPRESS • 38
 - UNIXCMD • 38
 - UNIXSH • 38
 - WAITOPER • 38
 - WAKEUP • 38
- message filtering • 25
- message records
 - assigning actions • 37
 - attributes • 37
 - routing to remote hosts • 38
 - types • 36
- messages
 - assigning actions • 37
 - attributes • 37
 - correlating • 48
 - distributing activity • 37
 - enhancement • 47
 - held • 50
 - input text • 36
 - log • 50
 - monitoring traffic • 49
 - output text • 36
 - remote hosts • 38

- routing to remote hosts • 38
- special handling • 36
- trapping • 34
- types • 36
- WTOR • 50

N

- NETSTAT • 181
- non-network configuration, CAICCI • 170
- NOTRA • 182

O

- outages, responding to • 250

P

- parallel sysplex configurations, defining • 190
- PARM statement • 29
- performance sizing, Agent Technology • 89
- PING • 182
- PROD • 105
- PROTSEC • 190

R

- RECYCLE • 182
- RELEASE • 183
- REMOTE statement, Windows • 177
- REPORT • 183
- resource initialization • 93
- REXX
 - binary conversion • 298
 - CAGSS implementation • 296
 - overview • 296

S

- scheduling, cross-platform • 200
- SNA network configuration, CAICCI • 186
- SNAP • 159
- SNMP
 - catrapd • 35
 - traps • 53, 158
- SNMP Monitor
 - environment variables • 160
 - explanation of • 158
 - IP address • 158
 - requirements • 158
 - stopping • 159
 - tracing • 159

STATUS • 183
STDENV • 29
SVCDUMP • 183
SYSDTECT • 184
sysplex, sharing CAIENF control options • 146
system authorization facility support • 120

T

TCP/IP
 protocol • 203
 specifying protocol parameters • 205
technical support, contacting
 starting • 263, 264
TRACE • 159
TRACEON • 184
TRCROUTE • 185
troubleshooting • 339

U

Unicenter CA-OPS/MVS integration • 30
user IDs • 289
USS environment • 346

V

visualization services • 21
VTAM communication
 communication routes • 264, 268
 configure • 267
 data transmission values • 264
 deactivate • 269
 starting • 264, 268

W

WAKE • 185

X

XCF communication
 communication routes, establishing • 264
 data transmission values • 264
 problems • 267
 starting • 263, 264
 subsystem names • 264
XNTREPORT • 185
XPS client • 200