CICS Transaction Gateway
Version 8 Release 1

UNIX and Linux Administration

SC34-7216-00

CICS Transaction Gateway
Version 8 Release 1

UNIX and Linux Administration

SC34-7216-00

Note
Before using this information and the product it supports, read the information in "Notices" on page 333.

This edition applies to Version 8.1 of the CICS Transaction Gateway for Multiplatforms program number 5724-I81 and CICS Transaction Gateway Desktop Edition program number 5725-B65 and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About this information

This information describes the planning, installation, configuration, and operation of the IBM® CICS® Transaction Gateway and the IBM CICS Transaction Gateway Desktop Edition products.

You should be familiar with the operating system on which CICS Transaction Gateway runs and also with Internet terminology.

This information is organized as shown in the following table:

Topic	Contents
"What's new in CICS Transaction Gateway V8.1" on page ix	Functional changes made in this version of the product.
../common/ctg_int.dita	Overview of the product and the functions it provides.
Chapter 2, "Planning," on page 9	Planning your installation, including the hardware and software you need to run the product.
Chapter 3, "Installing," on page 21	How to install the product.
"Configuring CICS server connections" on page 48	How to set up communication between CICS Transaction Gateway and CICS.
	How to configure the product.
Chapter 7, "Security," on page 157	How to set up the product to use the SSL network security protocol.
"Client security overview" on page 166	How to provide a user ID and password when connecting to a CICS server.
Chapter 8, "Performance," on page 173	How to tune the product, and other system components, to achieve the best possible performance.
"Operating the Gateway daemon" on page 193	How to operate the product.
Chapter 10, "Operating," on page 191	How to operate the Gateway daemon.
"Operating the Client daemon" on page 204	How to operate the Client daemon.
Chapter 11, "3270 terminal emulation and printing," on page 213	How to use the cicsterm and cicsprnt programs.
Chapter 12, "Resolving problems," on page 225	How to perform problem determination and problem solving.
Chapter 4, "Upgrading," on page 27	How to upgrade from an earlier version of the product.
Chapter 13, "Monitoring and statistics," on page 257	How to use the statistics and request monitoring exits provided by the product.

What's new in CICS Transaction Gateway V8.1

CICS Transaction Gateway includes enhancements in the areas of open integration, security and high availability.

Open integration

- Interoperability with 32-bit or 64-bit compilers and runtime environments is provided on Microsoft Windows using a pure .NET DLL. For more information see *Programming using the .NET Framework*.
- On Microsoft Windows, the .NET API is extended to provide CICS channels and containers support. For more information see *Using channels and containers in .NET programs*.
- The JCA 1.6 architecture is supported, enabling the use of JEE 6 certified applications servers, such as WebSphere® Application Server V8. For more information see *Supported JEE application servers*.
- AIX® V7 and Red Hat Enterprise Linux (RHEL) V6 platforms are now supported, along with CICS Transaction Server V4.2.
- An enhanced CICS Transaction Gateway plug-in is provided for the CICS Explorer® which supports user-defined connection groups, connection import and export capability, and column sorting. For more information see the *CICS Transaction Server V4.2 Information Center* at: `http://publib.boulder.ibm.com/infocenter/cicsts/v4r2/topic/com.ibm.cics.ts.productoverview.doc/concepts/Explorer.html`.

Security

- Security is enhanced through the use of password phrases with ECI and ESI requests over IPIC connections to CICS TS V4.2. For more information see *IPIC connection security*.
- Password management is now available from .NET applications and remote C applications using the ESI API. For more information see *Making ESI calls from .NET programs* and *Making ESI V2 calls in remote mode*.

High availability

- Dynamic server selection is available through a user exit in the Gateway daemon. The exit typically decides which CICS server to select based on the CICS server name, user ID, and transaction ID that accompany the request. For more information see *High availability*.

Additional features

- Transaction tracking is supported for ECI V2 and .NET client applications through their APPLID and APPLID qualifier. For more information see *Monitoring and statistics*.
- Origin data is available for remote Java client applications. For more information see *Monitoring and statistics*.

User information

- Message help is provided for the CICS Transaction Gateway administration program **ctgadmin**. For more information see *Viewing message help*.

- Reference tables "Which API can be used?" on page 17 and "Which protocol can be used?" on page 16 are included in the CICS Transaction Gateway information center.
- An improved message search capability is included in the CICS Transaction Gateway information center.

Removed and changed function

- The named pipe protocol for Windows is no longer supported. For more information see *Upgrading from Version 8 Release 0*.
- The JAR file cicsj2ee.jar file has been renamed to cicsjee.jar.
- The CICS Transaction Gateway .NET API has been upgraded. For more information see *CICS Transaction Gateway .NET applications*.
- The logical server has been deprecated and is superseded by policy-based dynamic server selection. For more information see *Logical CICS server definitions*.

Chapter 1. Overview

CICS Transaction Gateway for Multiplatforms

CICS Transaction Gateway for Multiplatforms is a high-performing, secure, and scalable connector that enables client applications in different runtime environments to access CICS servers.

Using standards-based interfaces, CICS Transaction Gateway delivers access to new and existing CICS applications. CICS Transaction Gateway also provides flexible deployment options for different architectures. An example is shown in Figure 1.

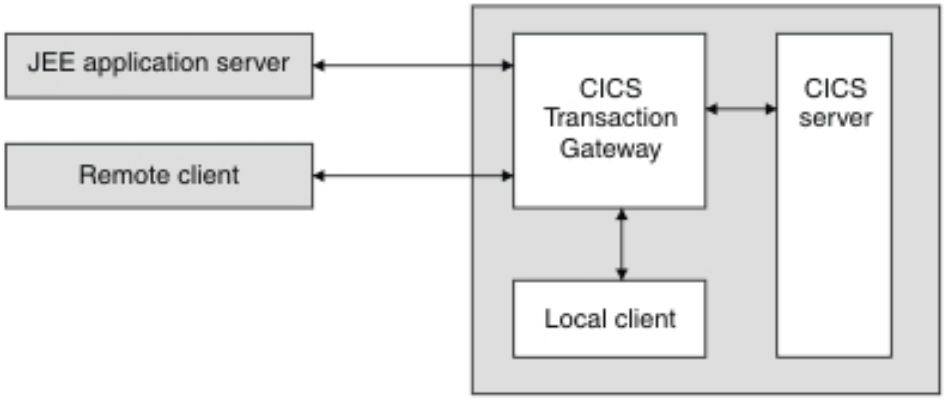


Figure 1. Access to CICS using CICS Transaction Gateway for Multiplatforms

On all operating platforms, CICS Transaction Gateway provides a gateway to CICS for remote clients, and complements IBM WebSphere Application Server on a range of different platforms. CICS Transaction Gateway for Multiplatforms also provides these features and benefits:

- A simple programming model requiring minimal change to CICS programs
- Access to CICS COMMAREA, channel and 3270 applications
- A rich set of client application programming interfaces (APIs) for different runtime environments
- Support for standard network protocols
- Support for different operating platforms
- Managed qualities of service and, on z/OS®, high availability
- Access to statistics and request monitoring information
- Support for two-phase commit transactions from a JEE application server

CICS Transaction Gateway Desktop Edition

CICS Transaction Gateway Desktop Edition provides single user desktop connectivity to CICS applications from a wide variety of client environments.

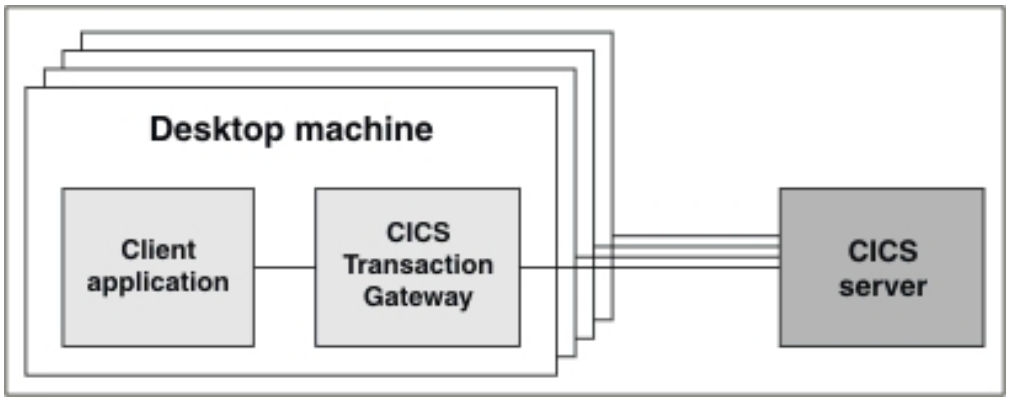


Figure 2. Access to CICS using CICS Transaction Gateway Desktop Edition

Client applications can be written in Java, .NET, C, C++, COM, or COBOL, and communicate with CICS Transaction Gateway Desktop Edition using either the local TCP/IP stack or interprocess communication. CICS Transaction Gateway Desktop Edition offers these features and benefits:

- A simple programming model requiring minimal change to CICS programs
- Access to CICS COMMAREA, channel and 3270 applications
- A rich set of client application programming interfaces (APIs) for different runtime environments
- Support for standard network protocols
- Support for different operating platforms
- Access to statistics and request monitoring information

CICS Transaction Gateway Desktop Edition does not include Java Connector Architecture (JCA) resource adapter support, and some parameters have restrictions. For more information about the parameter restrictions see:

- "Maximum requests" on page 85
- "Maximum number of worker threads" on page 71
- "Maximum number of connection manager threads" on page 70
- "IPIC send sessions" on page 53

Related information:

Chapter 2, “Planning,” on page 9
When planning a CICS Transaction Gateway installation, you must ensure that the requisite system hardware is available for running the product. You must also check that you have the correct software (for example, the correct operating system, web browser, CICS system and JEE application server). Finally, you must ensure that you use the correct communications protocols and interfaces for connecting to CICS on the platform on which CICS has been installed.

Application programming interfaces (APIs)

The application programming interfaces provide access to CICS COMMAREA programs, CICS channels and containers programs, and 3270 programs. APIs are included for the Java, C, C++, and COBOL programming languages. JCA resource adapters, are also included.

External Call Interface (ECI)

The ECI enables client applications to send requests to CICS COMMAREA and channel programs.

The ECI is available in all supported runtime environments. ECI is the most commonly used mechanism for providing client access to CICS. An ECI request results in a CICS distributed program link (DPL) call to the target program and must follow the CICS rules of the DPL subset.

JEE applications using the ECI resource adapter can access CICS resources as part of a two-phase commit transaction.

External Presentation Interface (EPI)

The EPI enables client applications to access CICS 3270-based programs. Client applications can install and delete virtual 3270 terminals in CICS through this interface. The EPI can be used in all supported runtime environments.

Basic mapping support (BMS) and non-BMS based terminal transactions are supported. Automatic transaction initiation (ATI) is supported.

External Security Interface (ESI)

The ESI enables client applications to call CICS password expiry management (PEM) functions. Client applications can access information about user IDs that are held in the CICS External Security Manager (ESM) through this interface.

Statistics API

The statistics API enables applications to obtain dynamic, real-time statistical information about the runtime performance of CICS Transaction Gateway. Applications can be written in C or Java.

Sample applications written in the supported programming languages are provided for all programming interfaces. For more information about working with the APIs, see the *CICS Transaction Gateway: Application Programming Guide*.

Programming Guide

Programming information for CICS Transaction Gateway including information on APIs, ancillary functions, user applications, and supported programming languages.

Programming

Deployment topologies

CICS Transaction Gateway can be deployed in a local mode (two-tier) topology or a remote mode (three-tier) topology. Each topology provides different qualities of service.

Connectivity options to CICS

There is a choice of network protocols for connecting to CICS.

All protocols support ECI COMMAREA requests.

IPIC This protocol is required for ECI channel requests and supports ESI requests. The protocol also supports two-phase commit transactions in local mode. SSL can be configured on IPIC connections in a local mode configuration.

SNA This protocol is required for sending EPI to CICS TS servers and supports ESI requests.

TCP/IP
This protocol is required for sending EPI requests to TXSeries servers.

Related information:

"Configuring CICS server connections" on page 48
After you have installed the CICS Transaction Gateway and set up your CICS servers for communication, your next step is to set up the communication links between the CICS Transaction Gateway and your CICS servers.

"Which protocol can be used?" on page 16
This table shows what support is available for connecting to different version CICS servers over IPIC, TCP/IP and SNA.

Remote mode

The client application and CICS Transaction Gateway are on different machines and the Gateway daemon listens on a specific port for incoming client requests. The Gateway daemon runs as a standalone process, handles the management of connections and threads, and forwards client requests to CICS.

In a remote mode configuration, the CICS Transaction Gateway runs a process known as a Gateway daemon which receives requests from client applications and forwards those requests to CICS servers. Client applications send requests to a Gateway daemon using either TCP/IP or SSL.

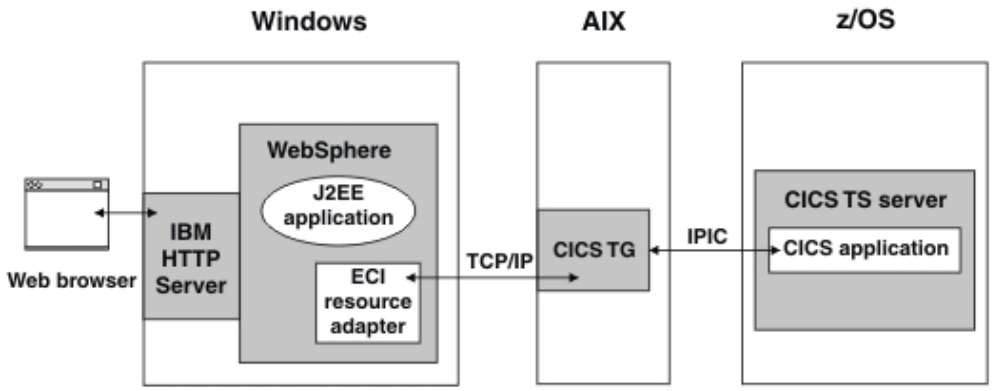


Figure 3. An example of CICS Transaction Gateway for Multiplatforms in remote mode

Features of remote mode

Remote mode is best suited to large-scale systems and has the following features:

- A common point of access to CICS for different applications and operating systems
- A common point of configuration and administration for connections to CICS
- A lightweight client footprint
- Access to statistical information
- Support for 64-bit JEE application servers
- Supports the use of applets to connect to a Gateway daemon

Local mode

In a local mode configuration, the CICS Transaction Gateway runs within the client application. Client applications send requests directly to CICS without using a Gateway daemon.

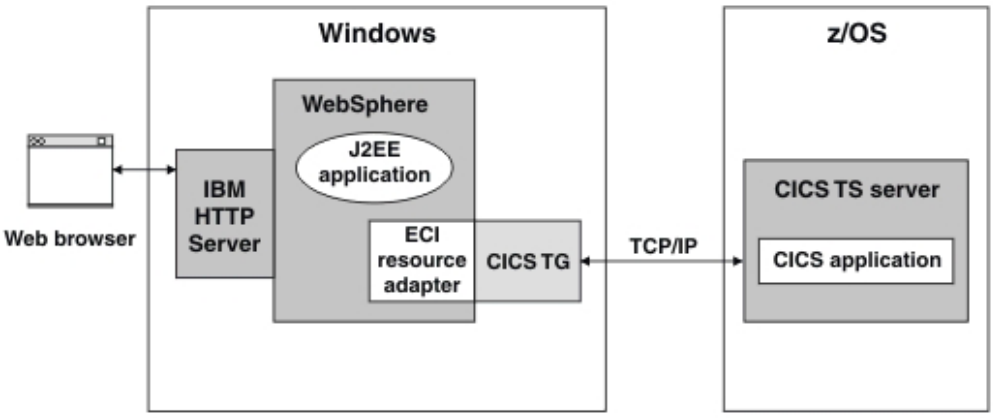


Figure 4. An example of CICS Transaction Gateway for Multiplatforms in local mode

Features of local mode

Local mode is best suited to environments where a small number of JEE application servers are connected to CICS, and has the following features:

- Fewer components to manage than in remote mode
- Network topology is simplified

High availability

High availability ensures that a single point of failure does not cause failure of the total solution. High availability also allows increased capacity to be provided by the addition of more components.

A high availability scenario can be implemented using dynamic server selection from within a user exit program.

Security

CICS Transaction Gateway provides a secure way of connecting to CICS using standard security mechanisms. These mechanisms integrate with security provided by the JEE application server and with security provided by CICS.

Network security

Network security is the ability to provide authentication and encryption over a network connection using these security technologies:

- Secure Sockets Layer (SSL) or Transport Layer Security (TLS) from a Java client application to CICS Transaction Gateway
- SSL or TLS from a Java client application to a CICS server using IPIC
- Security exits

Underlying security technologies such as Internet Protocol Security (IPSec) are also supported.

User authentication

User authentication is the process by which a service verifies a user's authenticity. Verification is through the use of credentials, usually a password or a certificate. User authentication can be implemented for all protocols.

Link security

Link security prevents a remote user from attaching to a transaction in CICS, or accessing a resource for which the link user ID has no authority. Link security provides an additional check on user authentication through the use of a preset user ID on the CICS server connection. Link security can be implemented for the SNA and IPIC protocols.

Bind security

Bind security prevents an unauthorized remote system from connecting to CICS. Bind security can be implemented for the SNA, TCP/IP, and IPIC protocols.

Statistics and monitoring

CICS Transaction Gateway provides statistics on the performance of runtime components. Monitoring information on individual requests is also available.

Statistics

The information provided by statistics is used when performing the following tasks:

- Capacity planning, where information about the resource usage is collected to ensure adequate capacity is available
- Hosting services and billing, where information on resource usage enables company or interdepartmental billing
- Runtime information, where a runtime "snapshot" of the system is used to evaluate status or perform high-level problem diagnosis

Statistics are retrieved by issuing local system administration commands, by using the C or Java statistics API, or by using third-party tools. The statistics API provides remote access from any platform.

Monitoring

Monitoring provides information about individual requests as they are processed by CICS Transaction Gateway. The information collected during monitoring includes:

- Key timestamps as a request passes through the CICS Transaction Gateway
- The client where each request originated
- The target CICS server for each request
- Request parameters such as the transaction identifier and program identifier
- The amount of data sent and received on each request
- Request tracking tokens

Monitoring is available through the use of user exit programs written in Java. Sample request monitoring exits are supplied.

Tooling and product integration

CICS Transaction Gateway is closely integrated with tools for application development and system monitoring.

- IBM Rational® Application Developer provides a J2C toolkit. The toolkit enables you to generate code for use with the CICS Transaction Gateway ECI resource adapters that JEE applications use when accessing CICS programs.
- IBM CICS Explorer provides access to CICS Transaction Gateway runtime statistics along with information from other CICS environments. The information is displayed in integrated views that can be customized.
- IBM Tivoli® OMEGAMON® automatically detects and provides alerts if critical transactions are not completed.

Chapter 2. Planning

When planning a CICS Transaction Gateway installation, you must ensure that the requisite system hardware is available for running the product. You must also check that you have the correct software (for example, the correct operating system, web browser, CICS system and JEE application server). Finally, you must ensure that you use the correct communications protocols and interfaces for connecting to CICS on the platform on which CICS has been installed.

For information about upgrading from an earlier version of CICS Transaction Gateway, see Chapter 4, "Upgrading," on page 27.

Hardware requirements

These are the machines, systems and processor types that support CICS Transaction Gateway.

- IBM System z® machine supported by Linux
- 32-bit or 64-bit IBM System p® supported by IBM AIX or Linux
- 32-bit or 64-bit Sun SPARC system supported by Sun Solaris
- 64-bit HP Itanium system supported by HP-UX
- Intel Pentium, AMD Opteron or Intel EM64T system supported by Linux

Supported software

CICS Transaction Gateway can be used with a range of operating systems, Web browsers, CICS systems and JEE application servers.

For the latest details about supported software, visit: Supported software for CICS Transaction Gateway products.

We recommend that the latest maintenance is applied to any supported software being used with CICS Transaction Gateway, according to the vendor's documentation.

Supported operating systems

CICS Transaction Gateway is supported on the listed operating systems.

AIX

CICS Transaction Gateway is supported on:

- AIX 7.1
- AIX 6.1
- AIX 5.3

On AIX V6.1 and AIX V7.1, application WPARs (workload partitions) are supported for remote Client applications, and for local Client applications connected to CICS over IPIC. CICS Transaction Gateway can run in shared system WPARs, and in private system WPARs. WPAR mobility is currently not supported.

Linux

CICS Transaction Gateway is a 32-bit application and is supported running on 32-bit operating systems and on 64-bit operating systems that can run 32-bit applications in native mode, leading to the following requirements for Linux:

- The 32-bit ncurses-lib package must be installed for running the Gateway daemon.
- For application development in C, the 32-bit glibc-devel Linux package must be installed and for C++ development, the 32-bit glibc-devel and libstdc++ Linux packages must be installed.

All UNIX and Linux operating systems require the Korn shell to be installed before you install CICS Transaction Gateway.

Note: CICS Transaction Gateway does not support Security-Enhanced Linux.

Linux on Intel

CICS Transaction Gateway is supported on:

- SuSE Linux Enterprise Desktop 11
- SuSE Linux Enterprise Desktop 10
- SuSe Linux Enterprise Server 11
- SuSe Linux Enterprise Server 10
- Red Hat Enterprise Linux Desktop 6
- Red Hat Enterprise Linux Desktop 5
- Red Hat Enterprise Linux 6
- Red Hat Enterprise Linux 5

Linux on POWER®

CICS Transaction Gateway is supported on:

- SuSE Linux Enterprise Server 11
- SuSE Linux Enterprise Server 10
- Red Hat Enterprise Linux 6
- Red Hat Enterprise Linux 5

Linux on System z

CICS Transaction Gateway is supported on:

- SuSE Linux Enterprise Server 11
- SuSE Linux Enterprise Server 10
- Red Hat Enterprise Linux 6
- Red Hat Enterprise Linux 5

Solaris on SPARC

CICS Transaction Gateway is supported on:

- Solaris 10

HP-UX on Itanium

CICS Transaction Gateway is supported on:

- HP-UX 11i V3
- HP-UX 11i V2

For the latest information on supported software see Supported software for CICS Transaction Gateway products.

Web browsers

CICS Transaction Gateway supports any web browser that can run applets at Java 5 or higher.

For the latest information on supported software see Supported software for CICS Transaction Gateway products.

Java support for the Gateway daemon

Java runtime environment support for CICS Transaction Gateway.

Service Release 9 is the minimum service level required and is shipped with CICS Transaction Gateway. You are recommended to use the latest Service Release level.

CICS Transaction Gateway supports the following Java runtime environments:

- IBM 32-bit Runtime Environment for AIX, Java Technology Edition, Version 6.0.
- IBM 32-bit Runtime Environment for Linux on Intel, Java Technology Edition, Version 6.0.
- IBM 32-bit Runtime Environment for Linux on iSeries® and pSeries®, Java Technology Edition, Version 6.0.
- IBM 32-bit Runtime Environment for Linux on zArchitecture, Java Technology Edition, Version 6.0.
- IBM 32-bit Runtime Environment for J2SE HP-UX 11i platform adapted by IBM for IBM software, Java Technology Edition, Version 6.0.
- IBM 32-bit Runtime Environment for Solaris, Java 2 Technology Edition, Version 6.0.

For the latest information on supported software see Supported software for CICS Transaction Gateway products.

Java support for Java Client applications

CICS Transaction Gateway supports these Java software development kits and runtime environments for use with Java Client applications.

Windows

- IBM Runtime Environment for Windows, Java Technology Edition, Version 6.0
- IBM Runtime Environment for Windows, Java Technology Edition, Version 5.0

AIX

- IBM Runtime Environment for AIX, Java Technology Edition, Version 6.0
- IBM Runtime Environment for AIX, Java Technology Edition, Version 5.0

Linux

- IBM Runtime Environment for Linux on Intel architecture, Java Technology Edition, Version 6.0
- IBM Runtime Environment for Linux on Intel architecture, Java Technology Edition, Version 5.0
- IBM Runtime Environment for Linux on iSeries and pSeries, Java Technology Edition, Version 6.0
- IBM Runtime Environment for Linux on iSeries and pSeries, Java Technology Edition, Version 5.0
- IBM Runtime Environment for Linux on zArchitecture, Java Technology Edition, Version 6.0
- IBM Runtime Environment for Linux on zArchitecture, Java Technology Edition, Version 5.0

Solaris

- IBM Runtime Environment for Solaris, Java Technology Edition, Version 6.0
- IBM Runtime Environment for Solaris, Java Technology Edition, Version 5.0

HP-UX

- HP Runtime Environment for J2SE HP-UX 11i platform, adapted by IBM for IBM Software, Version 6.0
- HP Runtime Environment for J2SE HP-UX 11i platform, adapted by IBM for IBM Software, Version 5.0

Notes

1. 32-bit and 64-bit Java Runtime Environments are supported.
2. Use of 64-bit Java in local mode is supported only when using IPIC to connect to CICS.
3. Use the latest Java update for your Java Runtime Environment.

For the latest details about supported software, visit: Supported software for CICS Transaction Gateway products.

Supported CICS servers

CICS Transaction Gateway is supported by these CICS servers.

- CICS Transaction Server for z/OS V4.2
- CICS Transaction Server for z/OS V4.1
- CICS Transaction Server for z/OS V3.2
- CICS Transaction Server for z/OS V3.1
- CICS V/SE 2.3
- CICS Transaction Server for VSE/ESA V1.1.1
- CICS Transaction Server for i5/OS® V6.1
- CICS Transaction Server for i5/OS V5.4
- CICS Transaction Server for i V7.1
- TXSeries for Multiplatforms V7.1
- TXSeries for Multiplatforms V6.2
- TXSeries for Multiplatforms V6.1

For the latest details about supported software,visit: Supported software for CICS Transaction Gateway products.

Restrictions on CICS Transaction Server for iSeries

Some restrictions apply to the use of CICS Transaction Gateway with CICS Transaction Server for iSeries.

The following restrictions apply:

- DBCS languages are supported when communicating using ECI but not EPI.
- Sign-on capable terminals are not supported.
- You cannot start the CEDA transaction from a client terminal.
- You cannot use PF1 to obtain CICS online help from a client terminal.

Supported JEE application servers

The resource adapters supplied with CICS Transaction Gateway are supported by a range of JEE application servers.

CICS Transaction Gateway resource adapters are supported in 64-bit application servers when using remote mode or when using local mode and the IPIC protocol. A 64-bit resource adapter used in remote mode can communicate with CICS Transaction Gateway running in 32-bit mode. Application servers are not supported by CICS Transaction Gateway Desktop Edition.

WebSphere Application Server

- WebSphere Application Server V8.0
- WebSphere Application Server V7.0 or earlier, only when used in remote mode with resource adapters from Supportpac CC03

Other JEE application servers

The following application servers are supported if they successfully run the JCA resource adapter installation verification test (IVT):

- JEE 6 certified application servers
- J2EE 1.4 or JEE 5 certified application servers, only when used in remote mode with resource adapters from Supportpac CC03

For more information about the IVT see "JCA resource adapter installation verification test (IVT)" on page 121.

For the latest details about supported software, visit: Supported software for CICS Transaction Gateway products.

Java base classes in J2EE application servers

Some restrictions apply to the base class APIs when used with servlets in the Web container.

The CICS Transaction Gateway base class APIs (JavaGateway, ECIRequest, EPIRequest) are no longer supported within the EJB container. The base class APIs are supported for usage with servlets in the Web container with the following limitations:

- All ECI requests must be non-transactional, meaning that only the field ECI_NO_EXTEND is supported on the ECIRequest constructor as the Extend_Mode.

- All ECI requests must be synchronous; that is, only the fields ECI_SYNC or ECI_SYNC_TPN are supported as the call types.
- The EPIRequest class is not supported within an application server. Use the EPI support classes (Terminal, Screen, and Field) instead.

Supported SNA communications products

To use SNA communications with CICS Transaction Gateway you must install one of these products.

AIX

- IBM Communications Server for AIX V6.4.0
- IBM Communications Server for AIX V6.3.0

Linux

- IBM Communications Server for Linux on zSeries® V6.4.0
- IBM Communications Server for Linux on zSeries V6.2.1
- IBM Communications Server for Linux V6.4.0
- IBM Communications Server for Linux V6.2.1

For the latest details about supported software, visit: Supported software for CICS Transaction Gateway products.

Supported compilers and application development tools

CICS Transaction Gateway supports a range of compilers and application development tools.

Windows

CICS Transaction Gateway supports the following products:

- Microsoft Visual Studio 2010
- Microsoft Visual Studio 2008
- Microsoft .NET framework 4.0
- Microsoft .NET framework 3.5
- IBM COBOL for Windows V7.6

AIX

CICS Transaction Gateway supports the following products:

- XL C/C++ for AIX V11.1
- XL C/C++ for AIX V10.1
- XL C/C++ Enterprise Edition for AIX, V9.0
- IBM COBOL for AIX V4.1
- IBM COBOL for AIX V3.1

Linux

CICS Transaction Gateway supports the following products:

- XL C/C++ for Linux, V11.1.0
- XL C/C++ for Linux, V10.1.0
- XL C/C++ Advanced Edition for Linux, v9.0

- gcc 4.1, 4.3, 4.4 for Linux

Solaris

CICS Transaction Gateway supports the following products:

- Oracle Solaris Studio 12.2
- Sun Studio 12.1
- Sun Studio 12
- Sun Studio 11

HP-UX

- ANSI C compiler for HP-UX
- aC++ compiler for HP-UX

Other supported compilers

32-bit ANSI compliant COBOL compilers, for example Micro Focus, are supported by the CICS Transaction Gateway C APIs.

For the latest details about supported software, visit: Supported software for CICS Transaction Gateway products.

GPL licence and copyright issues on Linux

This product uses the following libraries from the glibc package: libnsl.so, libm.so, libdl.so, ld.so, libc.so and libpthread.so.

Refer to the glibc package on your machine for the various copyright statements and licensing terms for these libraries.

This product also uses libncurses.so from the ncurses package. Again, refer to this package on your machine for the copyright statement and licensing terms applicable to this library.

IMPORTANT: Your use of the libstdc++ or egcs-c++ packages is subject to the GNU GPL licence terms which could require you to provide source code in certain circumstances. Note that IBM will not supply source code, for example for the CICS Gateways C++ libraries.

Virtualization

CICS Transaction Gateway supports virtualization solutions that can be implemented either by the hardware, or by the operating system.

Hardware-based virtualization solutions

If CICS Transaction Gateway is running on one of the supported operating systems, hardware-based virtualization can be provided by a hypervisor (virtual machine manager). CICS Transaction Gateway supports the following hypervisors:

- IBM Processor Resource/System Manager (PR/SM™) hypervisor with IBM z/OS or Linux operating systems
- IBM z/VM® hypervisor with z/OS or Linux operating systems
- IBM PowerVM™ hypervisor with IBM AIX or Linux operating systems
- VMware ESX Server with Windows or Linux operating systems

- Red Hat KVM with Red Hat Enterprise Linux (RHEL) operating system

Operating system-based virtualization solutions

If CICS Transaction Gateway for Multiplatforms is running on AIX V6.1 or later, operating system-based virtualization is available through workload partitioning (WPAR).

For information about the types of application and topologies that can be used in a WPAR virtualization environment, and the operating system versions that support WPAR see "Supported operating systems" on page 9.

Dynamic logical partitioning on AIX

CICS Transaction Gateway is a DLPAR-safe application that does not fail as a result of Dynamic Logical Partitioning (DLPAR) operations.

If resources are removed performance might be reduced, and if new resources are added scaling might not be possible, but the product still works as expected.

CICS Transaction Gateway does not manage I/O devices such as network adapters; before removing such devices dynamically, ensure that they are no longer being used by CICS Transaction Gateway or any of its underlying network components.

Which protocol can be used?

This table shows what support is available for connecting to different version CICS servers over IPIC, TCP/IP and SNA.

To determine which connectivity scenarios are supported by CICS Transaction Gateway, you should use this table in conjunction with the table "Which API can be used?" on page 17.

Table 1.

Header	IPIC IPv6	IPIC IPv4	TCI/IP IPv6	TCP/IP IPv4	SNA
CICS Transaction Server for z/OS V4.2	XA ECI + channels, ESI, password phrase	XA ECI + channels, ESI, password phrase	ECI	ECI	ECI, EPI, ESI
CICS Transaction Server for z/OS V4.1	XA ECI + channels, ESI	XA ECI + channels, ESI	ECI	ECI	ECI, EPI, ESI
CICS Transaction Server for z/OS V3.2	Not supported	ECI + channels	Not supported	ECI	ECI, EPI, ESI
CICS Transaction Server for z/OS V3.1	Not supported	Not supported	Not supported	ECI	ECI, EPI, ESI
CICS Transaction Server for VSE V1.1 and CICS/VSE V2.3	Not supported	Not supported	Not supported	ECI	ECI, EPI, ESI

Table 1. (continued)

Header	IPIC IPv6	IPIC IPv4	TCI/IP IPv6	TCP/IP IPv4	SNA
TXSeries V6.0 and V6.1	Not supported	Not supported	Not supported	ECI, EPI	Not supported
TXSeries V7.1	Not supported	ECI + channels	Not supported	ECI, EPI	Not supported
CICS Transaction Server for i5/OS V5.4 and V6.1	Not supported	Not supported	Not supported	ECI, EPI	ECI, EPI, ESI
CICS Transaction Server for i V7.1	Not supported	Not supported	ECI, EPI	ECI, EPI	ECI, EPI, ESI

Notes®:

- ECI denotes ECI COMMAREA application support.
- EPI denotes EPI API, CICS 3270 terminal emulator and CICS 3270 terminal printer support.

Which API can be used?

This table shows which APIs are supported over the IPIC, TCP/IP and SNA protocols in local and remote mode.

To determine which scenarios are supported by CICS Transaction Gateway, you should use this table in conjunction with the table in "Which protocol can be used?" on page 16

API	IPIC local mode	IPIC remote mode	TCP/IP local mode	TCP/IP remote mode	SNA local mode	SNA remote mode
Java ECI	✔ (see Note 1)	✔ (see Note 1)	✔	✔	✔	✔
Java ESI	✔	✔	x	x	✔	✔
Java EPI	x	x	✔	✔	✔	✔
JEE non-XA	✔ (see Note 1, Note 2)	✔ (see Note 1)	✔	✔	✔	✔
JEE XA	✔ (see Note 1, Note 2)	x	x	x	x	x
C/C++ ECI V1	x	x	✔	x	✔	x
C/C++ ESI V1	x	x	x	x	✔	x
C/C++ EPI	x	x	✔	x	✔	x
C ECI V2	x	✔ (see Note 1)	x	✔	x	✔
C ESI V2	x	✔	x	x	x	✔
.NET ECI	x	✔ (see Note 1)	x	✔	x	✔
.NET ESI	x	✔	x	x	x	✔

Notes:

1. Denotes channel application support. All ECI APIs and protocols support COMMAREA based applications.
2. Denotes support for SSL over IPIC. This is available only when running in local mode.

Compatibility

CICS Transaction Gateway provides a high level of interoperability between components, enabling applications, Gateways and CICS systems to be easily upgraded without the need for extensive changes.

For more information on CICS server compatibility see the CICS support page at: *http://www-01.ibm.com/software/htp/cics/ctg/support/*

Application compatibility

Compatibility of different versions of CICS Transaction Gateway Client applications with the Gateway daemon and when recompilation is required.

Java client application compatibility

Compatibility of Java client applications with different versions of the CICS Transaction Gateway API.

You do not have to recompile Java client applications if you migrate them to a new environment, for example if you:

- Upgrade the JVM on the client system
- Use a different operating system
- Update a remote CICS Transaction Gateway to a higher version
- Change the topology from local mode to remote mode

You must use a JRE version that is supported by the version of the ctgclient.jar that you have deployed with your Java client application.

C application compatibility

Compatibility of C applications with different versions of the CICS Transaction Gateway API.

C client applications do not have to be recompiled to run with a newer version of CICS Transaction Gateway unless there is a specific requirement to do so for the version of the product you are upgrading from.

If you already have ECI V2 applications deployed, and you upgrade your CICS Transaction Gateway to a later level, you can continue to use the existing version of ctgclient.dll on the remote machines or you can choose to upgrade it. You should consider the following points when choosing whether to upgrade ctgclient.dll on remote client machines:

- Is the latest maintenance required?
- Is the existing ctgclient.dll still at a supported level?
- Does the Client application connect to multiple CICS Transaction Gateways? (ECI V2 Client applications cannot connect to back-level Gateways).

Statistics application compatibility

Compatibility of statistical applications with different versions of the CICS Transaction Gateway API.

C and Java statistics applications do not have to be recompiled to run with a newer version of CICS Transaction Gateway. Statistics applications built using a particular version of CICS Transaction Gateway can connect to both newer and older version Gateway daemons.

If you already have a statistics application deployed, and you upgrade your CICS Transaction Gateway to a later level, you can continue to use the existing version of ctgclient.dll for C applications, or ctgstats.jar for Java applications on the remote machines, or you can decide to upgrade. You should consider the following points when deciding whether to upgrade ctgclient.dll or ctgstats.jar on remote client machines:

- Is the latest maintenance required?
- Is the existing ctgclient.dll or ctgstats.jar still at a supported level?

For Java applications, you must use a JRE version supported by the version of the ctgstats.jar that you have deployed.

User exit program compatibility

Compatibility of user exit programs with different versions of the CICS Transaction Gateway API.

CICS request exit and request monitoring exit programs do not have to be recompiled, if you upgrade CICS Transaction Gateway, and if the programs will execute with the version of Java required by CICS Transaction Gateway.

Client API C exit programs do not have to be recompiled if you upgrade CICS Transaction Gateway unless there is a specific requirement to do so for the version of the product you are upgrading from.

Resource adapter compatibility

Compatibility of different version resource adapters with different versions of CICS Transaction Gateway.

CICS Transaction Gateway can facilitate communications with CICS through resource adapters that are at the same version as CICS Transaction Gateway or at an earlier version. If you are migrating CICS Transaction Gateway to a later version, you can optionally migrate the earlier version resource adapters to the same level as CICS Transaction Gateway but this is not mandatory.

The following rules apply when using different versions of CICS Transaction Gateway and resource adapters:

- You can use a remote Gateway daemon with earlier version resource adapters.
- You cannot use a remote Gateway daemon with later version resource adapters.
- You cannot use a local Gateway with a different version resource adapter.
- You cannot mix earlier and later versions of resource adapters on the same JEE application server node.

Note: To find out which version number a resource adapter has, see the information provided by the application server for the resource adapter version. For example if you are using WebSphere Application Server, use the Administration Console to view the deployment descriptor for the installed resource adapter.

Code page support

Information about the code page support provided for CICS clients and CICS servers.

Server code page support

Some CICS servers do not support all the code pages that are supported by the operating system on which CICS Transaction Gateway is running.

If the code page of an ECI application is different from the code page of the server, data conversion must be performed at the CICS server. This restriction applies for EBCDIC CICS servers such as CICS Transaction Server for z/OS. For more information, see "Data conversion," on page 285, the CICS server documentation, and the IBM Redbooks® publication *Java Connectors for CICS*.

DBCS multibyte characters

Some characters in certain code pages are represented with 3 or more bytes. The CICS Transaction Gateway does not support multibyte characters that are longer than 2 bytes. If you try to display such characters on a CICS terminal, you will get unpredictable results.

If you are running on a locale that is unique to AIX or Solaris, you might experience problems when connecting to certain CICS servers. The following table shows the client and server combinations.

CICS Client code page	CICS Server operating system	CICS Server code page
ja_JP (33722)	AIX	932
ja_JP (33722)	HP	932
ja_JP (33722)	Linux	932
ja_JP (33722)	Solaris	932
ja_JP (33722)	Windows	932
ko_KR (970)	AIX	949
ko_KR (970)	HP	949
ko_KR (970)	Linux	949
ko_KR (970)	Solaris	949
ko_KR (970)	Windows	949
zh_TW (964)	AIX	950
zh_TW (964)	HP	950
zh_TW (964)	Linux	950
zh_TW (964)	Solaris	950
zh_TW (964)	Windows	950
zh_CN (1383)	AIX	1381
zh_CN (1383)	HP	1381
zh_CN (1383)	Linux	1381
zh_CN (1383)	Solaris	1381
zh_CN (1383)	Windows	1381

Chapter 3. Installing

Use the supplied SMP/E installation tape to transfer the CICS Transaction Gateway code to your system, following the supplied program directory.

Preparing to install CICS Transaction Gateway

Various tasks must be completed before installing CICS Transaction Gateway.

1. Check that your operating system is supported; for more information see "Supported software" on page 9.
2. Log on as root. Check the root user's umask to ensure that files created during the installation are readable by the user IDs that run CICS Transaction Gateway. A umask of 077 restricts access to the root user that installed CICS Transaction Gateway; a umask of 022 allows all users to run CICS Transaction Gateway.
3. On Linux operating systems issue the following command:

   ```
   umask 0022
   ```
4. Stop any programs that are running, for example the Client daemon if upgrading from V7.2 or earlier (issue the CICSCLI -x command), the Gateway daemon, or the Configuration Tool.
5. The installation process can upgrade from the CICS Transaction Gateway V6.0 or later. Remove any other versions of the CICS Transaction Gateway before installing this product. Remove any beta versions of the product.
6. The installation process requires 200 MB of free space for temporary data. By default, the installer uses the /tmp directory for the temporary data, or /root if there is insufficient space in /tmp. You can override this behavior by setting the *IATEMPDIR* environment variable before launching the installer, by issuing the command:

   ```
   export IATEMPDIR=/tempdir
   ```

   The installer then uses the directory specified in *IATEMPDIR* to store the temporary data.

Location of product files

The term <install_path> is used in file paths to represent the directory where you installed the product.

The location of the installed files for new installations of CICS Transaction Gateway is:

UNIX platforms

```
/opt/IBM/cicstg
```

Linux platforms

```
/opt/ibm/cicstg
```

Installing CICS Transaction Gateway

You can perform an attended install of CICS Transaction Gateway either from the graphical user interface or from the command line. Alternatively you can perform an unattended install, either with or without a response file.

Installing from the graphical user interface

When installing from the graphical user interface, it is not possible to cancel the installation process if it has already started to copy files.

To install from the graphical user interface:

1. Insert the DVD into the drive. If you are installing from a network drive, connect to the drive.
2. Navigate to the `ctg/`*`platform`* directory, where *`platform`* is your operating system.
3. Double-click on the `installer` file to launch the installation program.
4. Follow the on-screen instructions to complete the GUI installation.

Installing from the command line

When installing from the command line, it is not possible to cancel the installation process if it has already started to copy files.

To install from the command line:

1. Insert DVD into the drive. If you are installing from a network drive, connect to the drive.
2. Issue a command like the following:

   ```
   /DVD/platform/installer -i console
   ```

   where *`DVD`* is the name of your mounted DVD drive and *`platform`* is your operating system.
3. Follow the on-screen instructions to complete the installation.

Unattended installation using a response file

Unattended installation runs from the information stored in a response file; no user input is required.

To perform an unattended install using a response file:

1. Create a response file in one of the following ways:
   - Use the supplied sample file `installResponseSamp.txt`.
   - You can create a response file using the -r option, this option creates a file in the current working directory called `installer.properties`. To generate a response file issue the command **`installer -r`**.
2. Make the response file and the installation image available to the computer on which you are installing CICS Transaction Gateway.
3. Issue the following command:

   ```
   installer -i silent -f /tmp/installer.properties >/tmp/installer.log 2>&1
   ```

If there is a file named `installer.properties` in the same location as the installer executable file, it is used as a response file even if it is not specified on the command line.

As part of the installation symbolic links are created in `/usr/bin` and `/usr/lib` so that the product can run without any operating system configuration.

Unattended installation without a response file

To perform an unattended install without using a response file:

```
installer -i silent -DLICENSE_ACCEPTED=true >/tmp/installer.log 2>&1
```

Installing a supported JVM

The CICS Transaction Gateway requires a supported version of Java. You can configure the Gateway to use the Java supplied with the installer or you can install Java independently of CICS Transaction Gateway.

You can install a supported version of Java from CICS Transaction Gateway DVD, or as supplied in the product download files.

Uninstalling CICS Transaction Gateway

You can uninstall CICS Transaction Gateway either from the graphical user interface or from the command line. Alternatively you can perform an unattended uninstall either with or without a response file. The default uninstallation procedure is the same as the installation procedure.

Uninstalling from the graphical user interface

To uninstall CICS Transaction Gateway, run the `ctguninst` script as the root user:

```
ctguninst -i gui
```

Uninstalling from the console

To uninstall from the consolecommand line:

```
ctguninst -i console
```

Unattended uninstall using a response file

To perform an unattended install using a response file:

1. Copy and modify the supplied sample file `uninstallResponseSamp.txt` to a temporary file, for example: `/tmp/responsefile.txt`
2. Issue the following command:

   ```
   ctguninst -i silent -f /tmp/responsefile.txt >/tmp/uninstall.log 2>&1
   ```

Unattended uninstall without a response file

To perform an unattended uninstall without a response file (preserves the configuration, log, and trace files):

```
ctguninst -i silent >/tmp/uninstall.log 2>&1
```

To perform an unattended uninstall without using a response file (removes all files, including configuration, log, and trace files):

```
ctguninst -i silent -DPRESERVE_CONFIG_1=0 >/tmp/uninstall.log 2>&1
```

Location of the installation logs

Errors and warnings generated during the installation of CICS Transaction Gateway are recorded in the appropriate log file.

The log files, if created, are at the following locations:

Installation

- /installlogs/cicstgInstall.log
- /installlogs/cicstgMessages.log
- /installlogs/cicstgSymlinks.log
- /installlogs/cicstgFind.log

If the installation fails or the install is cancelled

- /tmp/cicstgInstall.log
- /tmp/cicstgMessages.log
- /tmp/cicstgSymlinks.log
- /tmp/cicstgFind.log

If an existing CICS Transaction Gateway could not be removed automatically during the upgrade the log below might also be created:

- /tmp/cicstgUninstall.log

If a log file already exists, it is overwritten.

Use the log to diagnose any problems that might have caused an installation to fail, especially in unattended mode, which has no user interaction with the program.

Redistributable components

You can develop C and .NET applications that access CICS servers using the CICS Transaction Gateway.

The ctgredist package is located in the <install_path>/deployable directory and provides components for developing C and .NET applications that access CICS servers using the CICS Transaction Gateway. Runtime libraries are provided for application deployment on systems remote to the CICS Transaction Gateway.

For information on installing the ctgredist package, see the ctgredist.txt file in the <install_path>/deployable directory.

CICS Transaction Gateway Desktop Edition: Support is not provided.

Using X-Window System from a remote system

The X-Window System software and a network protocol together provide a graphical user interface (GUI) for networked computers.

When using X-Window System from a remote system, for example, to access the Configuration Tool, you must set up the DISPLAY environment variable to allow the application to display its windows on that system.

On the display system (that is the one that will display the windows), enter the command:

```
xhost +appl
```

where *appl* is the network name of the system being used to run the application.

On the application system, before you run the application, enter the command:

```
export DISPLAY=disp:0.0
```

where *disp* is the host name or IP address of the system where the windows will be displayed (followed by a colon and the display id—normally 0.0). The application windows are then displayed on the *disp* system.

Chapter 4. Upgrading

When upgrading to a newer version of CICS Transaction Gateway, you should consider issues such as the installation directory location, whether your old version configuration files will still work, and whether all the old version parameters are still supported.

Upgrading from Version 8 Release 0

Considerations when upgrading from CICS Transaction Gateway Version 8 Release 0.

Using the JEE interfaces in nonmanaged mode

The JAR file cicsj2ee.jar file has been renamed to cicsjee.jar.

Upgrading from Version 7 Release 2

This information describes the issues to consider when upgrading from CICS Transaction Gateway Version 7 Release 2.

Installation directory

When you upgrade, the installation process keeps your existing <install_path>.

Configuration files

CICS Transaction Gateway configurations that worked with previous releases might not work after upgrade; configuration checking has been improved to ensure that the values used by CICS Transaction Gateway are the intended ones. Protocol handlers are not started unless explicitly configured.

Parameters:

- The GATEWAY section parameters **noinput** and **quiet** are no longer supported on Windows. Instances of these parameters that are found in a configuration file from a previous release, or as an override parameter, are ignored and a log message is produced.
- The GATEWAY section parameters **ecigenericreplies**, **msgqualvalidation** and **uowvalidation** are only supported when their default value is set, no message is generated for the unsupported value. The supported values are:

```
ECIGenericReplies=off
MsgQualValidation=on
UOWValidation=on
```

Java Version 6.0

Ensure that the PATH environment variable contains the location of the IBM Java 6.0 runtime environment, or set the CTG_JAVA environment variable to the full path of the IBM Java 6.0 launcher program. If you use ctgd to launch the Gateway daemon, set these variables in the configuration file (ctgd.conf).

Upgrading from Version 7 Release 1

The issues to consider when upgrading from CICS Transaction Gateway Version 7 Release 1.

Installation directory

When you upgrade, the installation process keeps your existing <install_path>.

Configuration files

CICS Transaction Gateway configurations that worked with previous releases might not work after upgrade; configuration checking has been improved to ensure that the values used by CICS Transaction Gateway are the intended ones. Protocol handlers are not started unless explicitly configured.

Parameters:

- The GATEWAY section parameters **noinput** and **quiet** are no longer supported on Windows. Instances of these parameters that are found in a configuration file from a previous release, or as an override parameter, are ignored and a log message is produced.
- The GATEWAY section parameters **ecigenericreplies**, **msgqualvalidation** and **uowvalidation** are only supported when their default value is set, no message is generated for the unsupported value. The supported values are:

```
ECIGenericReplies=off
MsgQualValidation=on
UOWValidation=on
```

Java Version 6.0

Ensure that the PATH environment variable contains the location of the IBM Java 6.0 runtime environment, or set the CTG_JAVA environment variable to the full path of the IBM Java 6.0 launcher program. If you use ctgd to launch the Gateway daemon, set these variables in the ctgd.conf configuration file.

Upgrading a statistics API port definition

If you have configured a statistics API port defined by a **statsport** parameter in the GATEWAY section of your configuration file, it is recommended that you upgrade to using a full statsapi protocol handler definition.

Previous releases of CICS Transaction Gateway bound the statistics API port exclusively to **localhost**. These monitoring applications were restricted to running on the same machine as the Gateway daemon. If you define a full statsapi protocol handler the remote monitoring applications can connect to the Gateway daemon. See “Statistics API protocol settings” on page 103 for details on remote statistics API connections.

Upgrading from Version 7 Release 0

This information describes the issues to consider when upgrading from CICS Transaction Gateway Version 7 Release 0.

Installation directory

When you upgrade, the installation process keeps your existing <install_path>.

Configuration files

CICS Transaction Gateway configurations that worked with previous releases might not work after upgrade; configuration checking has been improved to ensure that the values used by CICS Transaction Gateway are the intended ones. Protocol handlers are not started unless explicitly configured.

Parameters:

- The GATEWAY section parameters **noinput** and **quiet** are no longer supported on Windows. Instances of these parameters that are found in a configuration file from a previous release, or as an override parameter, are ignored and a log message is produced.
- The GATEWAY section parameters **ecigenericreplies**, **msgqualvalidation** and **uowvalidation** are only supported when their default value is set, no message is generated for the unsupported value. The supported values are:

```
ECIGenericReplies=off
MsgQualValidation=on
UOWValidation=on
```

Java Version 6.0

Ensure that the PATH environment variable contains the location of the IBM Java 6.0 runtime environment, or set the CTG_JAVA environment variable to the full path of the IBM Java 6.0 launcher program. If you use ctgd to launch the Gateway daemon, set these variables in the ctgd.conf configuration file.

Removal of TCP62 support

The releases of CICS Transaction Gateway V7.0 and CICS Universal Client V7.0 are the last releases that contained TCP62 support for the AnyNet® protocol communicating with remote CICS systems using SNA over TCP/IP protocol encapsulation. Accordingly, this capability has been removed from the V7.1 level of the products. For continued use of SNA over TCP/IP, it is necessary to move TCP62 server definitions to SNA and implement another IBM Communications Server TCP/IP protocol encapsulation solution, such as Enterprise Extender or Remote API client support.

For information about moving from TCP62 to Enterprise Extender support refer to the IBM publication *Migrating an SNA connection from TCP62 to Enterprise Extender - GC34-6889-00*.

For information about the removal of AnyNet support from z/OS Communications Server, refer to the z/OS and z/OS.e statements of direction announcement, Software Announcement 203-266, dated October 7, 2003, and the z/OS V1.7 preview announcement, Software Announcement 205-034, dated February 15, 2005.

Removal of SNA Server configuration setting LUALIASNAMES

LUALIASNAMES has been replaced by PARTNERLUALIAS.

If LUALIASNAMES is found in the configuration file after upgrade, the Client daemon attempts to start the SNA Server connection and generates a warning in

the Client daemon log file indicating that the deprecated entry LUALIASNAMES exists.

Upgrading from Version 6 and earlier

Version 6 and earlier cannot be upgraded; you must uninstall the earlier version before installing the later version.

Upgrading from CICS Universal Client

If you are upgrading from CICS Universal Client to CICS Transaction Gateway, you can use your CICS Universal Client configuration file with CICS Transaction Gateway.

No additional configuration is required unless you want to use features of CICS Transaction Gateway that were not supplied with CICS Universal Client. However, it is important to read the upgrading information for CICS Transaction Gateway to see whether any parameters you were using are not supported by the upgrade. You must also uninstall CICS Universal Client before installing CICS Transaction Gateway.

For more information see the link that matches the version of the CICS Universal Client from which you are upgrading:

- "Upgrading from Version 7 Release 0" on page 28
- "Upgrading from Version 7 Release 1" on page 28
- "Upgrading from Version 7 Release 2" on page 27

Chapter 5. Configuring

The configuration tasks needed to set up a CICS Transaction Gateway installation depend on factors such as network topology type, server connection type, and transaction type. Additional factors that determine which configuration tasks must be completed are the security, monitoring, and statics requirements.

Configuring the system environment

Perform these tasks to configure the system environment for CICS Transaction Gateway.

Set the JVM

If more than one JVM is installed on the machine you must specify which JVM CICS Transaction Gateway uses.

By default, CICS Transaction Gateway uses the first JVM defined on the PATH environment variable.

To specify which JVM CICS Transaction Gateway uses, set the CTG_JAVA environment variable.

For more information and an example, see "Environment variable reference" on page 34.

Set the time

Ensure that the locale and time zones are set correctly so that time stamps display the time accurately.

Changing the system locale

If you change the system locale you must change the language files.

To change the language in which user messages are displayed follow these steps:

1. Stop the CICS Transaction Gateway.
2. Change the locale of the machine to the locale in which messages are to be displayed. See your operating system documentation for information on how to do this.
3. Run the `ctgmsgs` command:

   ```
   ctgmsgs XX <code set>
   ```

   where *XX* is the two character message language. To obtain a list of available languages, enter the command without parameters.

```
This utility is used to change the language of user messages.
-------------------------------------------------------------

    Language               Locale          Code Set
    ---------------------  --------------  ----------
en  US English             en_US           ISO-8859-1
                           EN_US           UTF-8
fr  French                 fr_FR           ISO-8859-1
                           FR_FR           UTF-8
de  German                 de_DE           ISO-8859-1
                           DE_DE           UTF-8
it  Italian                it_IT           ISO-8859-1
                           IT_IT           UTF-8
es  Spanish                es_ES           ISO-8859-1
                           ES_ES           UTF-8
tr  Turkish                tr_TR           ISO-8859-9
                           TR_TR           UTF-8
ja  Japanese               ja_JP           EUCJP
                           JA_JP           UTF-8
ko  Korean                 ko_KR           EUCKR
                           KO_KR           UTF-8
zh  Simplified Chinese     zh_CN           EUCCN
```

Figure 5. Output from the ctgmsgs command

This also shows the locale and code set associated with the language.

4. Restart the CICS Transaction Gateway.

Using an alternative code set on AIX

On AIX issue the command **smitty iconv** to use an alternative code set. You can view a list of the alternative code sets using the smitty lang command, which is used to set the language environment.

This command provides the Convert Flat File screen, on which you can enter parameters:

```
                              Convert Flat File

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
* CURRENT FILE / DIRECTORY name                        []
* CURRENT CODE set                                     []                      +
* NEW FILE / DIRECTORY name                            []
* NEW CODE set                                         []                      +









F1=Help             F2=Refresh          F3=Cancel           F4=List
Esc+5=Reset         F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Fill in this screen as follows:

CURRENT FILE / DIRECTORY name
Enter <install_path>/bin/cclmsg.txt

CURRENT CODE set
Enter the code set for the language you selected with ctgmsgs.

NEW FILE / DIRECTORY name
Enter the name of the new file.

NEW CODE set
Enter the alternative code set. For example, the code set IBM-850 is an alternative for the US English code set ISO8859-1.

When you have done the conversion, overwrite the cclmsg.txt file with the new file.

Using an alternative code set on other operating systems

To use an alternative code set, use the **iconv** routine for the flat file <install_path>/bin/cclmsg.txt.

For example, to convert <install_path>/bin/cclmsg.txt from code set ISO8859-1 to code set ISO-850 enter:

```
iconv -f ISO8859-1 -t ISO-850 <install_path>/bin/cclmsg.txt > cclmsg.new
```

When you have done this conversion, overwrite the cclmsg.txt file with the new file:

```
mv cclmsg.new <install_path>/bin/cclmsg.txt
```

Configuring inter-process communication message queues

In UNIX and Linux systems, the Client daemon communicates internally using inter-process communication message queues. In systems other than AIX, the default configuration settings for these queues are too small to allow for large client data flows (such as 3270 maps or user COMMAREAs). Some symptoms of this problem are:

- An ECI program gives return code -3 (ECI_ERR_NO_CICS)
- A cicsterm locks when a large map is sent to it
- A large number of concurrent ECI requests significantly degrades performance

Change the configuration settings of the message queues to allow for large client data flows. The way that you do this depends on your operating system.

Message queues on HP-UX

The following settings are recommended:

```
msgssz  32     Message Segment Size
msgmnb  65535  Max Number of Bytes on Message Queue
msgmax  65535  Message Max Size (bytes)
msgseg  16384  Number of Segments Available for Messages
```

Set these values by using the **SAM** utility:

1. Type sam at the command prompt.
2. Select **Kernel Configuration**, **Configurable Parameters**.
   This displays a list of kernel parameters that you can change.
3. Select a parameter, either by clicking on it with the mouse or by moving the cursor to it and pressing the enter key.
4. Select **Actions**,**Modify configurable parameter**.
5. Enter the new value for the parameter in the **Formula/Value** field, and then select **OK**.

If the value you entered is not valid, SAM displays a window explaining the error.

6. When you have made all of the required changes, select **Actions**, **Process New Kernel**.

   SAM displays a window asking for confirmation; select **Yes**.

SAM then compiles the kernel and displays a window asking if you want to replace the old kernel before restarting the system. You must restart the system for the changes to take effect.

Message queues on Linux

You are recommended to place the following settings in file /etc/sysctl.conf.

```
kernel.msgmni=128               #Max # of msg queue identifiers
kernel.msgmnb=163840            #Size of message queue
kernel.msgmax=40960             #Max size of a message
```

- On Red Hat, the new settings are used after you restart the computer.
- On SuSE, issue the command `chkconfig boot.sysctl on`, and then reboot.
- To check that the new settings have been applied, issue the command `sysctl -a`.

The `MSGMNI` variable determines the maximum number of message queue identifiers system wide. This is typically set to 128 which is sufficient for the typical number of concurrent requests expected to be processed.

Message queues on Solaris

These settings are recommended for message queues on Solaris. You can set these values by changing the entries in the `/etc/system` file.

```
set msgsys:msginfo_msgmax = 65535  Maximum size of System V message.
set msgsys:msginfo_msgmnb = 65535  Maximum number of bytes that can be on any
                                   one message queue.
set msgsys:msginfo_msgssz = 32     Specifies size of chunks system uses to
                                   manage space for message buffers.
                                   Obsolete since the Solaris 8 release.
set msgsys:msginfo_msgseg = 16384  Number of msginfo_msgssz segments the system
                                   uses as a pool for available message memory.
                                   Total memory available for messages is
                                   msginfo_msgseg * msginfo_msgssz.
                                   Obsolete since the Solaris 8 release.
set semsys:seminfo_semmni = 4096   Maximum number of semaphore identifiers.
set msgsys:msginfo_msgtql = 10000  The maximum number of queue entries that
                                   can be in the system at the same time.
                                   A low value can adversely affect
                                   system performance, or cause the
                                   client to freeze. IBM recommends that
                                   you set this value to the maximum (10000),
                                   or at least double the maximum number of
                                   concurrent requests. Stress load your
                                   system, and then use the ipcs -qa command
                                   to determine the setting.
```

Environment variable reference

Environment variables, along with the configuration file, control how the CICS Transaction Gateway functions.

CICSCLI

Use the environment variable *CICSCLI* to specify an alternative configuration file for the product. For example,

```
export CICSCLI=/etc/cicstg/cicstg.ini
```

When *CICSCLI* is not defined the default name for the configuration file is:

<install_path>/bin/ctg.ini

CTGDCONF

Use the environment variable *CTGDCONF* to specify an alternative configuration file for the ctgd process. For example,

```
export CTGDCONF=/etc/cicstg/cicstgd.conf
```

When *CTGDCONF* is not defined the default name for the ctgd configuration file is:

```
<install_path>/bin/ctgd.conf
```

For more details on ctgd see, "Running the Gateway daemon as a background process" on page 196.

CTG_JAVA

Use the environment variable *CTG_JAVA* to specify the JVM used by the CICS Transaction Gateway. When you specify *CTG_JAVA* you must specify the full path and name of the Java launcher program. For example,

- AIX, Solaris and HP-UX

  ```
  export CTG_JAVA=/opt/IBM/cicstg/jvm160/bin/java
  ```

- Linux

  ```
  export CTG_JAVA=/opt/ibm/cicstg/jvm160/bin/java
  ```

If *CTG_JAVA* is not defined in the environment the PATH variable is used to find the Java launcher program.

Configuring a local mode topology

In a local mode topology, settings directly related to the Gateway daemon are not used. There are no thread pools or associated timeouts to configure. Requests are passed directly from Client applications to the CICS connectivity components.

To configure a local mode topology, complete the following tasks:

1. Configure the CICS server connections, as described in "Configuring CICS server connections" on page 48.
2. If you are using TCP/IP or SNA connections, configure the Client daemon settings, as described in "Configuring Client daemon settings" on page 84.
3. If you are using a Java client, set the PATH environment variable to include the <install_path>/bin subdirectory so that the application can find libctgjni.so.
4. If you are using a JEE application server, deploy the resource adapter that you require, as described in "Deploying the CICS resource adapters" on page 36.

Configuring a remote mode topology

Configuring remote Client application environments

The files required for compiling and running applications on a client machine are installed with CICS Transaction Gateway and must be copied to the client machine.

Java Client applications

The Java Virtual Machine (JVM) uses the CLASSPATH environment variable to find classes and zip or jar archives containing classes. To allow the JVM to access class files, specify the full path of directories containing class files or archives.

To compile and run Java applications on a client machine, add the full path of ctgclient.jar to the CLASSPATH environment variable. This archive is in the <install_path>/classes subdirectory. The JEE resource adapters are in the <install_path>/deployable subdirectory.

You must use a supported version of Java for running Java Client applications, a supported version of Java is provided on the CICS Transaction Gateway DVD, or as part of the product download.

C Client applications

The files required for compiling and running C applications on a client machine are in the ctgredist.tar.gz package in the <install_path>/deployable subdirectory. Copy the package to the client machine before extracting.

To compile ECI version 2 C applications on a client machine you must include the files ctgclient_eci.h, ctgclient.h and libctgclient in your C build environment. To run the applications the shared object libctgclient is required in the library path.

For information on building the supplied sample programs see the *CICS Transaction Gateway for Multiplatforms: Application Programming GuideCICS Transaction Gateway for z/OS: Application Programming Guide*.

Recommended Java options for the Solaris JVM

You are recommended to use the -XX:+UseLWPSynchronization Java option with Java Client applications.

Also ensure that /usr/lib/lwp appears before /usr/lib in the LD_LIBRARY_PATH variable.

If you do not set these options, calls to the JavaGateway.open() method might hang when either the TCP/IP or the SSL protocol is used. See your Solaris documentation for more details.

Deploying CICS TG applications

Deploying the CICS resource adapters

The resource adapters are provided as standard modules ready for deployment into a JEE application server. They can be packaged in JEE applications along with other components such as Enterprise Java Beans, and can be used to create larger, more complex systems.

CICS Transaction Gateway includes the following resource adapters which are located in the <install_path>/deployable directory:

- ECI resource adapter (cicseci.rar)
- EPI resource adapter (cicsepi.rar)

The resource adapters can be deployed in 32-bit and 64-bit runtime environments. For more information on supported environments see “Supported JEE application servers” on page 13.

For information on how to deploy the CICS resource adapters in a managed environment see your JEE application server documentation.

For more information about nonmanaged environments see the *CICS Transaction Gateway Programming Guide*.

If your JEE application server requires Java 2 Security permissions, or if you have enabled Java 2 Security permissions on your JEE application server, consider setting the security permissions that allow CICS Transaction Gateway to access your key stores. For more information see the *CICS Transaction Gateway Programming Guide*.

64-bit runtime environments

The CICS Transaction Gateway resource adapters can be used in local and remote modes with WebSphere Application Server running in a 64-bit runtime environment.

No additional configuration is required; the resource adapter automatically uses the correct library for the 64-bit runtime environment.

A 64-bit resource adapter used in remote mode can communicate with CICS Transaction Gateway running in 32-bit mode.

A 64-bit resource adapter used in local mode must use IPIC connections to CICS.

Transaction management models

CICS Transaction Gateway supports both the LocalTransaction and XATransaction transaction management models.

The xasupport custom property on a ConnectionFactory determines whether transactions use the XA protocol or not.

- To enable LocalTransaction support, set the xasupport custom property to `off`.
- To enable XATransaction support, set the xasupport custom property to `on`.

ECI resource adapter deployment parameters

The available deployment parameters for the ECI resource adapters and their effect on the final deployed resource adapter. The tools used to configure these parameters are server-specific. The default value is shown where appropriate. Parameters are optional unless indicated as required.

Applid
: In local mode, this parameter sets the APPLID used by the Client daemon and IPIC for CICS connections. In remote mode, this field is used to identify the client connection to the Gateway daemon.

ApplidQualifier
: In local mode, this parameter sets the APPLID QUALIFIER used by the Client daemon and IPIC for CICS connections. In remote mode, this field is used to identify the client connection to the Gateway daemon.

ConnectionURL
: The URL of the CICS Transaction Gateway instance with which the resource adapter will communicate. The URL takes the form `protocol://address`. This parameter is required. These protocols are supported:

  tcp

  ssl

  local

So, for example, in remote mode you might specify a URL of `tcp://ctg.business.com`. In local mode specify `local:`.

PortNumber
The port on which the Gateway daemon is listening. The default value for TCP/IP is 2006. This parameter is not relevant if you are running in local mode.

ServerName
The name of the CICS server to connect to for all interactions through this resource adapter. In remote mode, this name must be defined in the CICS Transaction Gateway configuration file. If this parameter is left blank, the default CICS server is used; For more information see "PRODUCT section of the configuration file" on page 114. To use multiple servers within an environment, you must deploy several Connection Factories, each with a different ServerName attribute. Each Connection Factory can use the same Resource Adapter. For an IPIC connection in local mode, this field specifies the server details as a URL: `protocol://hostname:port`.

SocketConnectTimeout
When connecting to a Gateway daemon in remote mode, this value is the maximum amount of time in milliseconds that the Java Client application allows for the socket to connect successfully.

When a Java Client application is running in local mode and communicating with a CICS server using the IPIC protocol, this value is the maximum amount of time that is allowed for the socket connection to CICS to happen successfully. If the Java Client application is using a protocol other than IPIC to communicate with the CICS server in local mode this value is ignored.

The default value of zero means that no timeout is applied when applicable.

TranName
The name of the CICS transaction under which you want all programs started by the resource adapter to run. The called program runs under a mirror transaction, but is linked to under the TranName transaction name. This name is available to the called program for querying the transaction ID.

Setting the TranName in the ECIInteractionSpec overrides the value as set at deployment (or on the ManagedConnectionFactory, if nonmanaged).

The TranName is equivalent to eci_transid. It does not affect the transaction under which the mirror program runs, but it can be seen in the exec interface block (EIB). When this option is used, the remote program runs under the default mirror transaction id CSMI, but the EIBTRNID field contains the eci_transid value.

TPNName
The name of the CICS TPN Transaction under which you want all programs started by the resource adapter to run. TPNName takes precedence if both TranName and TPNName are specified. If the TPNName is set on the ECIInteractionSpec, this setting overrides any values set at deployment time (or on the ManagedConnectionFactory, if nonmanaged).

The TPNName is equivalent to eci_tpn; it specifies a transaction under which the CICS mirror program runs. This option is like the TRANSID

option in an EXEC EXEC CICS LINK command. A transaction definition in CICS for this TRANSID must point to the DFHMIRS program.

UserName
The CICS user ID to be used if no other security credentials are available.

Password
The password for the CICS user ID specified in the UserName parameter.

ClientSecurity
The fully-qualified name of the ClientSecurity class to use in each interaction with CICS. This parameter is optional; if no value is given, no ClientSecurity class is used. For more information about the use of ClientSecurity classes and how to write them, see the information about CICS Transaction Gateway security classes in the *CICS Transaction Gateway for Multiplatforms: Programming Guide*.

ServerSecurity
The fully-qualified name of the ServerSecurity class to use in each interaction with CICS. This parameter is optional; if no value is given, no ServerSecurity class is used. For more information about the use of ServerSecurity classes and how to write them, see the information about CICS Transaction Gateway security classes in the *CICS Transaction Gateway for Multiplatforms: Programming Guide*.

KeyRingClass
The fully-qualified name of the SSL keystore to use. The use of this field depends on the type of connection from the resource adapter. If the resource adapter is making an IPIC connection directly to CICS (local mode), then KeyRingClass is the name associated with the IPIC connection. If the resource adapter is using a remote mode SSL connection to a Gateway daemon, then KeyRingClass is the name associated with the SSL connection.

KeyRingPassword
The password for the keystore defined in KeyRingClass.

TraceLevel
The level of trace to be output by the resource adapter. For more details on trace levels and tracing see “JEE Tracing” on page 253.

Cipher Suites
The "CipherSuites" can be used when establishing an SSL connection. In the WebSphere Administration console, change the "CipherSuites" custom property for the connection factory to a comma-separated list of the cipher suites that this connection factory is restricted to use.

RequestExits
A list of fully-qualified request monitoring exit class names delimited from each other by commas (","). Each class must implement the com.ibm.ctg.monitoring.RequestExit interface and be on the class path. For more information about the use of RequestExit classes and how to write them, see the information about Java request monitoring user exits in the *CICS Transaction Gateway for Multiplatforms: Programming Guide*.

In addition to these user-definable properties, the ECI resource adapters have a set of predefined attributes that each deployed resource adapter inherits. These properties are defined in the JEE/CA specification and are as follows:

Transaction support
The cicseci resource adapter's transactional support is defined as LocalTransaction.

Reauthentication support
The ECI resource adapters support reauthentication. Reauthentication is the ability to change the security credentials when a connection is requested from the server and an already existing one is allocated without having to disconnect and reconnect to the EIS. Reauthentication improves performance.

The ECI resource adapters have a set of predefined attributes that each deployed resource adapter inherits when in local mode connecting over IPIC. These attributes cannot be defined by the user.

Server Idle Timeout
Inactive connections to a CICS server are disconnected after 60 minutes.

Send TCP KeepAlive packets
Periodically send keepalive messages to the server to check the connection.

Send Sessions
The number of simultaneous transactions, or CICS tasks, that are allowed over the connection is determined by the IPCONN RECEIVECOUNT parameter in CICS Transaction Server for z/OS.

XASupport
When using this connection, the transaction type to be used. If this is set to off, Local transactions are used. If this is set to on, XA transactions are used.

Deployment parameters for the EPI resource adapter

The EPI resource adapter has the following deployment parameters. The tools used to configure these parameters are server specific. The default value is shown where appropriate. Parameters are optional unless indicated as required.

Applid
In local mode, this parameter sets the APPLID used by the Client daemon and IPIC for CICS connections. In remote mode, this field is used to identify the client connection to the Gateway daemon.

ApplidQualifier
In local mode, this parameter sets the APPLID QUALIFIER used by the Client daemon and IPIC for CICS connections. In remote mode, this field is used to identify the client connection to the Gateway daemon.

ConnectionURL
The URL of the CICS Transaction Gateway with which the resource adapter will communicate. The URL takes the form `protocol://address`. This parameter is required. These protocols are supported:

tcp

ssl

local

For example, you might specify a URL of `tcp://ctg.business.com`. For the local protocol specify `local:`.

PortNumber
The port on which the CICS Transaction Gateway is running. The default value is 2006. This parameter is not relevant if you are using the local protocol.

ServerName
The name of the CICS server to connect to for all interactions through this resource adapter. If this parameter is left blank, the default CICS server is used (see "Default server" on page 48). This name is defined in the CICS Transaction Gateway configuration file. To use multiple servers within an environment, you must deploy several Connection Factories, each with a different ServerName attribute. Each Connection Factory can use the same Resource Adapter.

SocketConnectTimeout
The maximum time in milliseconds that a Java Client application tries to open a socket connection to a remote Gateway daemon. The default value 0 means that no timeout is applied. The timeout is ignored for attempts to connect to a local Gateway.

UserName
The CICS user ID to be used if no other security credentials are available. See the information about Programming using the J2EE Connector Architecture in the *CICS Transaction Gateway for Multiplatforms: Programming Guide* for more information.

A LogonLogoff class is required if:

- The requested terminal is sign-on capable, or
- The CICS Server does not support sign-on capable terminals (for example, CICS Transaction Server for iSeries)

Password
The password for the CICS user ID defined above.

ClientSecurity
The fully qualified name of the ClientSecurity class to use in each interaction with CICS. This parameter is optional; if no value is given, no ClientSecurity class is used. For more information about the use of ClientSecurity classes, and how to write them, see the information about CICS Transaction Gateway security classes in the *CICS Transaction Gateway for Multiplatforms: Programming Guide*.

ServerSecurity
The fully qualified name of the ServerSecurity class to use in each interaction with CICS. This parameter is optional; if no value is given, no ServerSecurity class is used. For more information about the use of ServerSecurity classes, and how to write them, see the information about CICS Transaction Gateway security classes in the *CICS Transaction Gateway for Multiplatforms: Programming Guide*.

KeyRingClass
The fully qualified name of the SSL key ring to use. This applies only when using the SSL protocol.

KeyRingPassword
The password for the key ring defined in KeyRingClass. Because it is linked to KeyRingClass, it is also optional, and applies only to the SSL protocol.

SignonType
The EPI resource adapter allows you to define whether the CICS terminals used by the resource adapter are sign-on capable. Enter one of the following:

0 Sign-on capable terminal (default)

1 Sign-on incapable terminal

For information about sign-on capability, see “Sign-on capable and sign-on incapable terminals” on page 168.

Encoding
The Java Encoding to use when creating 3270 data streams. The encoding is converted to the appropriate CCSID. See “Supported conversions” on page 285 for a list of supported encodings. Ensure that your CICS Server supports the CCSID for the given encoding.

LogonLogoffClass
The fully-qualified name of the Java Class that provides the logic to log on to a CICS Server using sign-on transactions. This property is mandatory for sign-on capable terminals and for CICS servers that do not support sign-on capability (such as CICS Transaction Server for iSeries). See the information about writing LogonLogoff classes in the *CICS Transaction Gateway for Multiplatforms: Programming Guide* for information about LogonLogoff classes.

DeviceType
The Terminal Model type, as defined in CICS, to be used by this resource adapter.

ReadTimeout
The maximum time a CICS server waits for a response from a user or J2EE component when taking part in a conversational transaction. Values for this parameter are:

0 No timeout.

1– 3600
Time in seconds.

InstallTimeout
The timeout value for terminal installation on CICS. Values for this parameter are as follows:

0 No timeout.

1– 3600
Time in seconds.

TraceLevel
The level of trace to be output by the resource adapter. For more details on trace levels and tracing see “Tracing” on page 245.

Cipher Suites
The "CipherSuites"can be used when establishing an SSL connection. In the WebSphere Administration console, change the "CipherSuites" custom property for the connection factory to a comma-separated list of the cipher suites that this connection factory is restricted to use.

RequestExits
This field is not used by the EPIResourceAdapter and should be left blank.

In addition to these user definable properties, the EPI resource adapter has a set of predefined attributes that each deployed resource adapter inherit's. These properties are defined in the J2EE/CA specification and are as follows:

Transaction support
The EPI resource adapter is nontransactional. It can be used within transactional contexts, but does not react to commit or rollback requests.

Reauthentication Support
The EPI resource adapter supports re-authentication. Reauthentication is the ability to change the security credentials when a connection is requested from the server and an already existing one is allocated, without having to disconnect and reconnect to the EIS. Reauthentication improves performance. A LogonLogoff class is required if the terminal requested is SignonCapable, or if the CICS Server does not support sign-on capable terminals (such as CICS Transaction Server for iSeries).

Deploying ECI V2 and ESI V2 to remote systems

Remote ECI V2 and ESI V2 applications are deployed as executable files.

You are licensed to copy the following files to the machine that is running the ECI V2 and ESI V2 application:

ECI V2 and ESI V2 CICS TG API runtime library:

- ctgclient.dll (Windows)
- libctgclient (Unix and Linux)

You can find this file in the <platform>/lib directory of the ctgredist package or in the <install_path>\lib of an installed CICS TG.

At run time the ctgclient must be available on the system path or in the same directory as the ECI V2 and ESI V2 application.

CICS Transaction Gateway Desktop Edition: Support is not provided.

Deploying .NET applications to remote systems

Remote .NET applications are deployed to the Windows runtime environment as an executable (.exe) file.

You are licensed to copy the following file to the computer that is running the .NET application:

.NET CICS Transaction Gateway API assembly: IBM.CTG.Client.dll

The CICS Transaction Gateway .NET API supports Microsoft .NET Framework versions 3.5 and 4.0, and 32-bit and 64-bit Windows architectures. Support is provided by a single DLL (IBM.CTG.Client.dll) which is included in the ctgredist package in the directory Windows\lib or in <install_path>\lib on a Windows machine with CICS Transaction Gateway installed.

You must deploy IBM.CTG.Client.dll in the Global Assembly Cache, or in the same directory as the .NET application.

For further information on deploying assemblies in the Global Assembly Cache refer to the Microsoft documentation.

CICS Transaction Gateway Desktop Edition: Support is not provided.

Using the Configuration Tool

Use the Configuration Tool to configure CICS Transaction Gateway.

To use the Configuration Tool remotely, export the display using the commands described in "Using X-Window System from a remote system" on page 24. To start the Configuration Tool, enter the **ctgcfg** command.

Storing configuration details

Configuration details are stored by default in the ctg.ini file in the <install_path>. You are recommended to use the Configuration Tool to create and update this file.

You can specify a different location and optionally change the name of the configuration file ctg.ini by setting the CICSCLI environment variable. If this variable is set, at startup the Configuration Tool loads the file referenced by CICSCLI.

For example, if CICSCLI is set as `CICSCLI=/u/userid/myconfig.ini`, the Configuration Tool tries to load `/u/userid/myconfig.ini` and `/u/userid/ctg.env` at startup. The status bar shows which configuration file is being edited.

If the Configuration Tool fails to find the configuration files, it creates them; these files must be edited and saved before use. If one of the files exists it reads in the values from that file, and creates the missing file. It does not automatically create a directory if the directory referenced by CICSCLI does not exist.

At startup the name and location of the configuration file is written to the Gateway information log. CICS Transaction Gateway does not run if a configuration file is not found.

Mixed-case values

The configuration tool allows parameter values to be entered in mixed-case and also writes the values as mixed-case to the configuration file, ctg.ini.For both the Gateway daemon and the Client daemon, some values are folded to uppercase at run time.

Running the Configuration Tool for a different operating system

Proceed as follows:

1. Install the product on the workstation on which you want to run the Configuration Tool.
2. To edit an existing configuration, copy ctg.ini to the UNIX and Linux on the workstation, using FTP in ASCII mode.
3. Issue the following command to display help about the Configuration Tool:
   ```
   ctgcfg -?
   ```
4. Issue the following command:
   ```
   ctgcfg -PLAT OSCODE
   ```

   where *OSCODE* represents the operating system that the Configuration Tool should emulate, and is one of the values returned by the command that you issued at step 3.

5. Make the required entries and click **Save** to create or update the ctg.ini file.
6. Use FTP in ASCII mode to transfer the file to your system.

Identification using APPLID

CICS Transaction Gateway supports identification using APPLID. This provides a standard mechanism for identification of Gateway daemon and Java client components in the CICSplex, and for subsequent task correlation in CICS.

Gateway identification has implications for the following:

- Qualification of log messages
- IPIC connections to identify the Gateway daemon to CICS
- Request monitoring data
- TCP/IP connections to identify the Gateway daemon to CICS

APPLID qualifier and APPLID

A fully-qualified APPLID is formed from an APPLID qualifier string and an APPLID string separated by a period symbol:

```
<APPLID qualifier>.<APPLID>
```

Each part can be between 1 and 8 characters in length and is defined independently. In certain configurations the defining of a fully-qualified APPLID is mandatory. See “IPIC server connections” on page 46 for further details.

If the configuration file (ctg.ini) contains an APPLID but not an APPLID qualifier, the system uses the default value 9UNKNOWN for APPLID qualifier. For more information, see “Gateway APPLID qualifier.”

The APPLID parameter replaces the existing Client Application ID parameter. The Client Application ID parameter is supported for migration purposes, but is overridden by the new APPLID parameter.

Gateway APPLID

The **applid** parameter identifies the instance of the CICS Transaction Gateway on server connections and tasks in a CICSplex.

Set a value of up to 8 characters in the **APPLID** field of the Configuration tool. There is no restriction on the characters that can be used, however, to ensure that the APPLID is valid for all scenarios, use characters in the range A through Z, and 0 through 9. The value that you set is converted to uppercase for SNA and TCP/IP connections.

The value must be unique within the CICSplex. If you do not set a value, the system automatically generates a value that is guaranteed to be unique.

This parameter is in the PRODUCT section of the configuration file, see PRODUCT section of the configuration file for more information about other parameters in this section.

Gateway APPLID qualifier

The **applidqualifier** parameter is used as a high-level qualifier for the APPLID.

Set the value up to 8 characters in the **applidqualifier** field of the Configuration tool. There is no restriction on the characters that can be used, however, to ensure that the **applidqualifier** is valid for use in all scenarios, use characters in the range A through Z, and 0 through 9.

The default value is 9UNKNOWN.

The combination of **applid** and **applidqualifier** identifies CICS Transaction Gateway to the CICS system to which it connects.

If the configuration file (ctg.ini) contains an **applid** but not an **applidqualifier**, the system uses the default value 9UNKNOWN for **applidqualifier**. This value matches the initial default in CICS Transaction Server. If the default is kept, the value is included in messages generated in the Gateway daemon and in CICS, and in statistics. Having a default provides a reference value that makes problem diagnosis simpler. The default can be used in either a local mode or a remote mode topology.

This parameter is in the PRODUCT section of the configuration file, see "PRODUCT section of the configuration file" on page 114 for more information about other parameters in this section.

IPIC server connections

CICS Transaction Gateway IPIC connections in CICS are identified by a fully-qualified APPLID.

In remote mode set the fully-qualified APPLID to be used to identify CICS Transaction Gateway to CICS in the configuration file, and in local mode set the fully-qualified APPLID in the application or environment. If the APPLID and APPLID qualifier specified in an IPCONN in CICS match this APPLID and APPLID qualifier, the configuration of that IPCONN is applied to the connection made by the Gateway daemon.

If there is no matching IPCONN definition, the connection is autoinstalled if the CICS system has been configured to autoinstall IPCONN connections. If you configure CICS not to allow autoinstall of IPCONN connections, only requests that have APPLIDs that are set on the predefined IPCONN definitions are able to connect.

If the configuration file (ctg.ini) contains an APPLID but not an APPLID qualifier, the system uses the default value 9UNKNOWN for APPLID qualifier. For more information, see "Gateway APPLID qualifier" on page 45.

IPIC connections with a defined fully-qualified APPLID

If the Gateway daemon or local mode application is configured with a fully-qualified APPLID, and connects to a CICS server using the IPIC protocol, no other Gateway daemon or local mode application configured with the same fully-qualified APPLID can concurrently establish an IPIC connection with the same CICS server. If the fully-qualified APPLID is not unique, attempts made to connect to a CICS server might be rejected because another connection has already been installed using the same fully-qualified APPLID.

When the Gateway daemon or local mode application is configured with a fully-qualified APPLID, all application requests sent to a CICS server using the

IPIC protocol use that fully-qualified APPLID. There is no check of fully-qualified APPLIDs for uniqueness, so you must choose a naming convention carefully to ensure the uniqueness of fully-qualified APPLIDs across the enterprise.

IPIC connections without a defined fully-qualified APPLID

When a Gateway daemon or local mode application without a defined fully-qualified APPLID connects to a CICS server using the IPIC protocol the CICS server generates a fully-qualified APPLID that is unique to that connection. If this Gateway daemon or local mode application connects to multiple CICS servers using the IPIC protocol, each connections own fully-qualified APPLID is generated independently. Each time that a Gateway daemon or local mode application without a defined fully-qualified APPLID connects to a CICS server using the IPIC protocol, the fully-qualified APPLID that is generated changes and cannot be relied on to be consistent.

Establishing an IPIC Connection

If the APPLID and NETWORKID specified in a CICS IPCONN definition match the Gateway daemon or local mode application's APPLID and APPLID qualifier, the configuration of that IPCONN is applied to the connection made by the Gateway daemon. If there is no matching IPCONN definition, the connection is autoinstalled if the CICS system has been configured to allow autoinstall IPCONN connections. If you configure CICS to prohibit the autoinstall of IPCONN connections only requests that have APPLIDs that are set on the predefined IPCONN definitions can connect.

If the APPLID qualifier defined for the Gateway daemon or local mode application is left blank and the NETWORKID in the CICS IPCONN definition is left blank, a match will not occur even if the APPLIDs match, because CICS defaults the blank NETWORKID to the local network ID.

SNA and TCP/IP server connections

The Client daemon uses the APPLID value specified. If the APPLID is not set, and the deprecated Application ID parameter on the CLIENT section is set to anything other than “*”, this value is used.

If multiple Client daemon server connections are configured to a CICS server and you specify an APPLID, that name must be unique. If the name is not unique, attempts to connect to a server might be rejected because another connection has already been installed using the same APPLID.If the Client daemon wants to communicate with a given server over SNA, the APPLID might be overridden at the time the client is installed at the server by the Local LU name for the client. For TCP/IP connections, whether the APPLID is used to identify connections or not is server implementation dependant.

The APPLID parameter replaces the existing Client Application ID parameter. The Client Application ID parameter is supported for upgrade purposes, but is overridden by the new APPLID parameter.

The APPLID qualifier is not used for SNA and TCP/IP connections.

Configuring CICS server connections

After you have installed the CICS Transaction Gateway and set up your CICS servers for communication, your next step is to set up the communication links between the CICS Transaction Gateway and your CICS servers.

Default server

The **defaultserver** parameter is used for requests in which no CICS server name is specified.

The default server is not used by cicsterm or cicsprnt.

This parameter is in the PRODUCT section of the configuration file, see PRODUCT section of the configuration file for more information about other parameters in this section.

Configuring IPIC

Perform these steps to configure an IPIC server connection.

The TCP/IP stack on your local machine is typically already correctly configured. Contact your system administrator if you encounter problems.

An IPIC connection between CICS Transaction Gateway and CICS Transaction Server must not be load balanced through any TCP/IP port sharing or load balancing software.

IP interconnectivity (IPIC)

IPIC provides ECI access to CICS applications over the TCP/IP protocol, supporting both COMMAREA and CICS channel applications and two phase commit. CICS channels and containers allow you to send and receive more than 32 KB of application data in a single ECI request. IPIC cannot be used with the EPI or ESI interfaces.

Transactional support

IPIC supports two-phase commit XA transactions in local mode only.

For information about the transaction types supported by the IPIC protocol when using CICS Transaction Gateway to connect to different CICS servers see “Which protocol can be used?” on page 16

Connections to CICS

For IPIC communications between CICS Transaction Gateway and CICS Transaction Server V4.1 (or higher), up to two sockets are used for each IPIC connection. If the IPIC connection is defined to use a maximum of one session, a single socket is used.

For IPIC communications between CICS Transaction Gateway and CICS Transaction Server V3.2 or TXSeries systems, one socket is used for each IPIC connection.

If you lose one or more of the sockets in use by an IPIC connection, for example, because of a network error, all the sockets are lost and the IPIC connection is released.

Multiple sockets are automatically used for IPIC connections where they are supported by the connected system. The use of multiple sockets is indicated by a Gateway daemon log message, which is generated when the connection is established.

Related information:

"Which API can be used?" on page 17
This table shows which APIs are supported over the IPIC, TCP/IP and SNA protocols in local and remote mode.

"Which protocol can be used?" on page 16
This table shows what support is available for connecting to different version CICS servers over IPIC, TCP/IP and SNA.

"IPIC connection security" on page 159
IPIC connections enforce link security to restrict the resources that can be accessed over a connection to a CICS server, bind security to prevent an unauthorized client system from connecting to CICS, and user security to restrict the CICS resources that can be accessed by a user. If the CICS server supports password phrases, a password phrase can be used for user security.

Verifying the TCP/IP installation

Perform these steps to verify that CICS Transaction Gateway can communicate with CICS servers.

1. Use the TCP/IP PING command to check the route to the CICS server:

   ```
   ping [machine address | name]
   ```

   To start PING, enter a command like the following:

   ```
   ping 192.113.36.200
   ```

   where 192.113.36.200 is the IP address of your CICS server. If you are using a *Domain Name Server (DNS)*, you can specify the symbolic host name rather than the IP address of the server.
2. If the statistics message returned shows a value other than 0% packet loss, it is possible that TCP/IP is not correctly configured:
   - Check for TCP/IP definition errors.
   - Check the physical connection to the network.

The PING command differs slightly, depending on your operating system. For more information, refer to the documentation supplied with your operating system.

Configuring IPIC on CICS Transaction Server for z/OS

Perform these steps to configure IPIC on CICS Transaction Server for z/OS.

CICS Transaction Gateway can send IPIC requests over TCP/IP to CICS Transaction Server for z/OS V3.2 and later. To perform this configuration:

1. Set the System Initialization (SIT) parameter TCPIP=YES.
2. Define the TCP/IP address and host name for the z/OS system. By default, they are defined in the PROFILE.TCPIP and TCPIP.DATA data sets.
3. Add a TCP/IP listener to CICS. Use the following CEDA command to define a TCPIPSERVICE in a group:

   ```
   CEDA DEF TCPIPSERVICE(service-name) GROUP(group-name)
   ```

   Ensure that the group in which you define the service is in the startup GRPLIST, so that the listener starts when CICS is started. Key fields are explained as follows:

POrtnumber
The port on which the TCP/IP service listens.

PRotocol
The protocol of the service is IPIC.

TRansaction
The transaction that CICS runs to handle incoming IPIC requests. Set it to CISS (the default).

Backlog
The number of TCP/IP requests that are queued before TCP/IP starts to reject incoming requests.

Ipaddress
The IP address (in dotted decimal form) on which the TCPIPSERVICE listens. For configurations with more than one IP stack, specify ANY to make the TCPIPSERVICE listen on all addresses.

SOcketclose
Whether CICS waits before closing the socket after issuing a receive for incoming data on that socket. NO is recommended for IPIC connections, to ensure that the connection from the CICS Transaction Gateway always remains open.

4. Use the following command to install the TCPIPSERVICE definition:
   `CEDA INS TCPIPSERVICE(`*service-name*`) GROUP(`*group-name*`)`
5. Choose whether to predefine or to autoinstall IPIC connections in CICS Transaction Server for z/OS. Specific inbound connections can be defined for different configurations using the CICS definition, IPCONN, or the connection can be autoinstalled using either the default or a customized autoinstall program. When CICS TG connects to CICS it flows the fully-qualified APPLID defined for the Gateway daemon or local mode application and if this matches that defined on an IPCONN definition, that definition is used to install the connection. If there is no matching IPCONN definition, the connection is autoinstalled. For further information on setting the fully-qualified APPLID for IPIC connections see "IPIC server connections" on page 46.

   To customize autoinstalled IPIC connections, for example, to configure security, an IPCONN definition must be created with the customized attributes to act as a template and this definition must be referenced as the template in a customized IPCONN autoinstall user program. The name of the autoinstall user program must be specified on the URM option of the installed TCPIPSERVICE definition. For further information on setting security on IPIC connections see "IPIC connection security" on page 159.

   When creating an IPCONN definition for a CICS TG to CICS connection, the SENDCOUNT parameter must be set to zero, unlike CICS to CICS connections for which the SENDCOUNT must not be zero.

Setting session limits

The number of simultaneous transactions, or CICS tasks, that are allowed over the connection is determined as follows:

Table 2. The number of simultaneous transactions allowed over an IPIC connection

SENDSESSIONS setting in CICS Transaction Gateway	IPCONN Receive Count setting in CICS Transaction Server for z/OS	Number of simultaneous transactions allowed
Set	Set (on IPCONN resource definition or customized autoinstall)	The lesser of the two values is used.
Set	Not set (default autoinstall)	The value of the CICS Transaction Gateway SENDSESSIONS setting is used.
Not set	Set (on IPCONN resource definition or customized autoinstall)	The value of the CICS Transaction Server for z/OS IPCONN Receive Count setting is used.
Not set	Not set (default autoinstall)	A value of 100 is used.

Each active session uses one CICS task, so the maximum number of sessions allowed is 999. CICS Transaction Gateway allocates 300 KB of memory for each session. If all the defined sessions are in use, any new requests receive an ECI_ERR_RESOURCE_SHORTAGE error.

For more information on configuration file definitions for IPIC, see "IPICSERVER section of the configuration file" on page 119.

Configuring IPIC in local mode

For IPIC connections in local mode, the CICS server name (ServerName) is defined as a URL. A URL allows you to specify a protocol, host name, and port number, which is the minimum information you need to connect to CICS.

The URL has the following format:

```
Protocol://hostname:port
Protocol://hostname:port#CICSAPPLID
Protocol://hostname:port#CICSAPPLIDQUALIFIER.CICSAPPLID
```

where:

- *Protocol* is either tcp or ssl.
- *hostname* is the TCP address of the host.
- *port* is the port number of the TCPIPSERVICE listener in CICS.
- *CICSAPPLID* is the APPLID of the CICS server.
- *CICSAPPLIDQUALIFIER* is the network ID of the CICS server.

CICSAPPLID and CICSAPPLIDQUALIFIER are optional parameters. If specified, these parameters are sent to CICS when the connection is established and are validated by CICS. The connection is rejected if the CICSAPPLID and CICSAPPLIDQUALIFIER do not match the CICS server. If you do not specify the CICSAPPLID and CICSAPPLIDQUALIFIER parameters, no check is made.

Configuring IPIC in remote mode

In remote mode, the IPIC server definitions are stored in the configuration file (ctg.ini). If the incoming server name found in the configuration file refers to an IPIC definition, IPIC is used, otherwise the request is sent to the Client daemon for processing.

Configuring an IPIC CICS Server definition

Use the CICS Transaction Gateway configuration tool to configure a new IPIC CICS server definition, or edit the IPICSERVER section of the configuration file directly.

To configure a new IPIC CICS Server definition, using the Configuration Tool:

1. Select **New Server** from the **Options** menu, or from the toolbar, or right click the CICS Servers entry in the Navigation Panel.
2. Select a server in the navigation panel to display the Server connection panel. The settings map to the parameters in a Server section of the configuration file.

   Multiple identical server definitions are not permitted in the configuration file of the CICS Transaction Gateway. The combination of fields that identify an IPIC server are Hostname or IP address, Port, CICS APPLID and CICS APPLID qualifier.

   This ensures that every connection that the CICS server and the network protocol see is represented by a unique server definition in the configuration file.
3. Set the values required as described in the following sections.

To configure an IPIC server definition edit the configuration file directly, see "IPICSERVER section of the configuration file" on page 119 for more information.

Server name:

The **SECTION IPICSERVER** parameter provides a server name that is independent of the communications protocol for the server, local to the CICS Transaction Gateway.

Set the value in the range of 1 to 8 characters in the **Server name** field of the Configuration tool.

Supported characters are in the ranges A-Z and 0-9, and '@', '#', '$', '-'. Lower case characters you enter in the range a-z are converted to upper case. If you do not use single-byte characters, other characters might not be displayed correctly.

Use this name for all requests to access the server from Client applications.

This parameter is in the SERVER section of the configuration file, see "SERVER section of the configuration file" on page 119 for more information about other parameters in this section.

Description:

The **description** parameter specifies a description for the server.

Set the value in the range between 1 and 60 characters in the **Description** field of the Configuration Tool, in single-byte characters. Use single-byte characters to avoid other characters being displayed incorrectly. The value is optional.

The description is returned on list systems calls.

Depending on the configuration used, the value is defined in one of the following locations:

- "IPICSERVER section of the configuration file" on page 119
- "SERVER section of the configuration file" on page 119

Network protocol:

The **protocol** parameter specifies the protocol to be used for the server connection.

Select the protocol to be used for the server connection. This can be on of the following values:

- TCP/IP
- SNA
- IPIC

You must include a SECTION DRIVER for the protocols listed above, excluding IPIC, for example, SECTION DRIVER = SNA, and specify the correct **drivername** parameter. See "DRIVER section of the configuration file" on page 120 for more information.

If you use configuration tool, select the protocol to be used from the drop down list in the in the **Network protocol** field.

Host name or IP address:

The **hostname** parameters identifies the host on which the CICS is running.

Set the value as a character or numeric TCP/IP identifier, for the host on which CICS is running, for example, host name `cicssrv2.company.com` or IP Address `192.113.36.200`. This field is mandatory.

Host names are mapped to IP addresses either by the name server or in the hosts system file. It is better to use a host name in case the IP address changes.

The hosts system file is in the /etc directory.

This parameter is in the IPICSERVER section of the configuration file, see "IPICSERVER section of the configuration file" on page 119 for more information about other parameters in this section.

Port:

The **PORT** parameter defines the port number on which the target CICS server is listening.

Set the value in the range 1 and 65,535.

This parameter is in the IPICSERVER section of the configuration file. For more information about the parameters in this section see "IPICSERVER section of the configuration file" on page 119.

IPIC send sessions:

The **SENDSESSIONS** parameter specifies the number of simultaneous transactions or CICS tasks that are allowed over the CICS connection.

Set the value in the range 1 to 999.

Set the **SENDSESSIONS** in the CICS Transaction Gateway configuration file and the Receive Count IPCONN definition in CICS Transaction Server for z/OS.

If SENDSESSIONS is not set in the configuration file, CICS TG defaults to requesting 100 sessions. This might be negotiated down if RECEIVECOUNT is set to a lower number on the server, if set at all.

This parameter is in the IPICSERVER section of the configuration file, see "IPICSERVER section of the configuration file" on page 119 for more information about other parameters in this section.

CICS Transaction Gateway Desktop Edition: The maximum value for this parameter is 5.

Target CICS APPLID:

The **CICSAPPLID** parameter is optional unless the **CICSAPPLIDQUALIFIER** parameter is specified. If specified, the **CICSAPPLID** must match the APPLID of the target CICS server.

Set the value up to 8 characters in length, specifying the APPLID of the target CICS server.

The **CICSAPPLIDQUALIFIER** is optional unless **CICSAPPLID** is specified. If specified, the **CICSAPPLIDQUALIFIER** must match the network ID of the target CICS server. Set the alphanumeric value up to 8 characters in length, specifying the network ID of the target CICS server.

This parameter is in the IPICSERVER section of the configuration file, see "IPICSERVER section of the configuration file" on page 119 for more information about other parameters in this section.

Connection timeout:

The **CONNECTTIMEOUT** specifies the maximum time in seconds that establishing a connection is allowed to take.

Set the value in the range 0 to 3600. The default value of 0 means that no limit is set.

This parameter is in the IPICSERVER section of the configuration file, see "IPICSERVER section of the configuration file" on page 119 for more information about other parameters in this section.

Server retry interval:

The **srvretryinterval** parameter specifies the time in seconds between attempts by the Gateway daemon to reconnect to a CICS server over an IPIC connection.

Set the value between 0 and 3600. The default is 60 seconds. The server retry interval setting determines whether server re-connection attempts are made exclusively at fixed timed intervals, or are made exclusively as a result of ECI requests being directed to the server.

If the CICS server which is currently connected becomes inactive, an attempt is made to reconnect one second after the CICS server becomes inactive. If the connection attempt fails, additional attempts are made to connect at the interval specified by the **srvretryinterval** parameter.

Set the value to 0 to prevent automatic connection attempts. Each request directed at the CICS server initiates a connection attempt if one is not already in progress.

This parameter is located in the IPICSERVER section of the configuration file. For more information see "IPICSERVER section of the configuration file" on page 119.

Server idle timeout:

The **SRVIDLETIMEOUT** parameter specifies, in minutes, the period of inactivity after which the connection between the Gateway daemon and the CICS server is closed.

Set the value in the range 0 to 1080.

A value of 0 means that no timeout is applied. The default value is 60 minutes.

This parameter is in the IPICSERVER section of the configuration file, see "IPICSERVER section of the configuration file" on page 119 for more information about other parameters in this section.

Send TCP KeepAlive packets:

The **TCPKEEPALIVE** parameter periodically sends keepalive messages to the server to check the connection.

Set this value to **YES** if you want TCP/IP send keepalive messages to the server to check the connection.

The CICS Transaction Gateway uses the interval specified by your operating system.

The default is **YES**.

This parameter is in the IPICSERVER section of the configuration file, see "IPICSERVER section of the configuration file" on page 119 for more information about other parameters in this section.

Configuring TCP/IP

Perform these steps to configure a TCP/IP server connection.

The TCP/IP stack on your local machine should already be correctly configured. Contact your system administrator if you encounter problems.

Verifying the TCP/IP installation

Perform these steps to verify that CICS Transaction Gateway can communicate with CICS servers.

1. Use the TCP/IP PING command to check the route to the CICS server:

   ```
   ping [machine address | name]
   ```

   To start PING, enter a command like the following:

   ```
   ping 192.113.36.200
   ```

   where 192.113.36.200 is the IP address of your CICS server. If you are using a *Domain Name Server (DNS)*, you can specify the symbolic host name rather than the IP address of the server.
2. If the statistics message returned shows a value other than 0% packet loss, it is possible that TCP/IP is not correctly configured:

- Check for TCP/IP definition errors.
- Check the physical connection to the network.

The PING command differs slightly, depending on your operating system. For more information, refer to the documentation supplied with your operating system.

Configuring TCP/IP on CICS Transaction Server for z/OS

Perform these steps to configure TCP/IP on CICS Transaction Server for z/OS.

To perform this configuration:

1. Set the SIT parameter TCPIP=YES.
2. Install the following:
   - CICS-supplied transient data queue CIEO, in group DFHDCTG
   - Transaction CIEP in group DFHIPECI
   - Program DFHIEP in group DFHIPECI
3. Define the TCP/IP address and host name for the z/OS system. By default they are defined in the PROFILE.TCPIP and TCPIP.DATA data sets.
4. Add a TCP/IP listener to CICS. Use the following CEDA command to define a TCPIPSERVICE in a group:

   ```
   CEDA DEF TCPIPSERVICE(service-name) GROUP(group-name)
   ```

   Ensure that the group in which you define the service is in the startup GRPLIST, so that the listener starts when CICS is started. Key fields are explained as follows:

   POrtnumber
   The port on which the TCP/IP service listens.

   PRotocol
   The protocol of the service is ECI.

   TRansaction
   The transaction that CICS runs to handle incoming ECI requests. Set it to CIEP.

   Backlog
   The number of TCP/IP requests that are queued before TCP/IP starts to reject incoming requests

   Ipaddress
   The IP address (in dotted decimal form) on which the TCPIPSERVICE listens. For configurations with more than one IP stack, specify ANY to make the TCPIPSERVICE listen on all addresses.

   SOcketclose
   Whether CICS should wait before closing the socket after issuing a receive for incoming data on that socket. NO is recommended for ECI connections, to ensure that the connection from the Client daemon always remains open.

   ATtachsec
   Specifies the level of attach-time security required for TCP/IP connections.
5. Use the following command to install the TCPIPSERVICE definition:

   ```
   CEDA INS TCPIPSERVICE(service-name) GROUP(group-name)
   ```

For information about configuring TCP/IP on other CICS servers, refer to your server documentation.

Configuring a TCP/IP CICS Server definition

Use the CICS Transaction Gateway configuration tool to configure the a new TCP/IP CICS server definition, or edit the SERVER section of the configuration file directly.

To configure a new TCP/IP CICS Server definition using the Configuration Tool:

1. Select **New Server** from the **Options** menu, or from the toolbar, or right click on the CICS Servers entry in the Navigation Panel.
2. Select a server in the navigation panel to display the Server connection panel. The settings map to the parameters in a Server section of the configuration file.

   Multiple identical server definitions are not permitted in the configuration file of the CICS Transaction Gateway. The combination of fields that identify a TCP/IP server are Host name or IP address and Port.

   This ensures that every connection that the CICS server and the network protocol see is represented by a unique server definition in the configuration file. If you have existing Client applications that use different server names to send requests to the same CICS server, you need to write a user exit to redirect requests. This is demonstrated in sample user exits `ecix2.c` and `epix2.c`; for more information, see the *CICS Transaction Gateway for Multiplatforms: Application Programming GuideCICS Transaction Gateway for z/OS: Application Programming Guide*.
3. Set the values required as described in the following sections.

To configure a TCP/IP server definition by editing the configuration file directly, see "SERVER section of the configuration file" on page 119.

Server name:

The **`SECTION SERVER`** parameter provides a server name that is independent of the communications protocol for the server, local to the CICS Transaction Gateway.

Set the value in the range of 1 to 8 characters in the **Server name** field of the Configuration tool.

Supported characters are in the ranges A-Z and 0-9, and '@', '#', '$', '-'. Lower case characters you enter in the range a-z are converted to upper case. If you do not use single-byte characters, other characters might not be displayed correctly.

Use this name for all requests to access the server from Client applications.

This parameter is in the SERVER section of the configuration file, see "SERVER section of the configuration file" on page 119 for more information about other parameters in this section.

Description:

The **`description`** parameter specifies a description for the server.

Set the value in the range between 1 and 60 characters in the **Description** field of the Configuration Tool, in single-byte characters. Use single-byte characters to avoid other characters being displayed incorrectly. The value is optional.

The description is returned on list systems calls.

Depending on the configuration used, the value is defined in one of the following locations:

- "IPICSERVER section of the configuration file" on page 119
- "SERVER section of the configuration file" on page 119

Network protocol:

The **protocol** parameter specifies the protocol to be used for the server connection.

Select the protocol to be used for the server connection. This can be on of the following values:

- TCP/IP
- SNA
- IPIC

You must include a SECTION DRIVER for the protocols listed above, excluding IPIC, for example, SECTION DRIVER = SNA, and specify the correct **drivername** parameter. See "DRIVER section of the configuration file" on page 120 for more information.

If you use configuration tool, select the protocol to be used from the drop down list in the in the **Network protocol** field.

Initial transaction:

The **initialtransid** parameter specifies a transaction identifier of between 1 and 128 characters.

This string is case-sensitive and identifies the initial transaction, and any parameters, to be run when the terminal emulator connects to the server. If you do not enter anything in the **Initial transaction** field of the Configuration tool, no initial transaction is run. The first four characters, or the characters before the first blank in the string are taken as the transaction. The remaining data is passed to the transaction on its invocation.

Ensure that the transaction does not require terminal input.

This parameter is in the SERVER section of the configuration file, see "SERVER section of the configuration file" on page 119 for more information about other parameters in this section.

Model terminal definition:

The **modelterm** parameter specifies the name of a model terminal definition at the server.

Set the value of the string between 1 and 16 characters in the **Model terminal definition** field of the Configuration tool. The string is case-sensitive and identifies the characteristics of terminals to be autoinstalled from the client. If the model cannot be located at the server, or you do not enter anything, a default terminal definition is used. This default is server-specific.

The interpretation of the Model terminal definition setting is server-specific. For example, for a TXSeries for AIX server, the value is 1 to 16 characters, and is the DevType for a CICS terminal definition entry to be used as the model.

This parameter is in the SERVER section of the configuration file, see "SERVER section of the configuration file" on page 119 for more information about other parameters in this section.

Host name or IP address:

The **netname** parameter is the character or numeric TCP/IP identifier for the host on which the CICS server is running.

For example enter, cicssrv2.company.com (host name) or 192.113.36.200 (IP Address) in the **Hostname or IP address** field of the Configuration tool.

Host names are mapped to IP addresses either by the name server or in the `hosts` system file. It is better to use a host name in case the IP address changes.

The `hosts` system file is in the /etc directory.

This parameter is in the SERVER section of the configuration file, see "SERVER section of the configuration file" on page 119 for more information about other parameters in this section.

Port:

The **port** parameter defines the port number at the server to which the Client daemon connects.

Set the value in the range 0 through 65,535 in the **Port** field of the Configuration tool. The default value is 0.

A value of 0 indicates that the services system file must be used to locate the port number for the CICS service using the TCP protocol. The services system file is in the /etc directory.If an entry cannot be discovered in the services system file, a value of 1435 is used (the number of the port assigned to the Client daemon in the TCP/IP architecture).

This parameter is defined in the SERVER section of the configuration file. For information about the other parameters in this section see "SERVER section of the configuration file" on page 119.

Connection timeout:

The **connecttimeout** parameter specifies the maximum time in seconds that establishing a connection is allowed to take.

Set the value in the range 0 through 3600, in the **Connection timeout (s)** field of the Configuration tool,the default value of 0 means that no limit is set by the Client daemon.

A timeout occurs if connection establishment takes longer than the specified time. The TCP/IP socket is closed and the return code passed back to the client application is either ECI_ERR_NO_CICS or CICS_EPI_ERR_FAILED.

Cleanup processing happens after a timeout only if the server has support for this function installed.

This parameter is in the SERVER section of the configuration file, see "SERVER section of the configuration file" on page 119 for more information about other parameters in this section.

Server idle timeout:

The **srvidletimeout** parameter specifies in minutes, the period of inactivity after which the connection between the Client daemon and the CICS server is closed.

Set an integer in the range 1 through 1080 in the **Server idle timeout (min)** field of the Configuration tool, a value of 0 means that no timeout is applied. The default value is 0.

The field is available if one of the following network protocols is selected:

- TCP/IP
- SNA
- IPIC

The **Server idle timeout (min)** period is counted from when the number of outstanding conversations (units of work) on the connection is zero. The connection is automatically reestablished when a Client application sends the next ECI, EPI or ESI request.

When a server connection times out, the Client daemon behaves as if the `cicscli -x=<servername>` command had been issued. In particular, any value specified in the "Server retry interval (Client daemon connections to CICS)" on page 87 configuration field is ignored, and no attempt is made to reestablish the connection. The Server retry interval setting is re-enabled if the connection is re-established.

If the network protocol for the server in question is not supported for this field, the Client daemon ignores any entry in the configuration file for the **Server idle timeout (min)** field.

This parameter is in the SERVER section of the configuration file, see "SERVER section of the configuration file" on page 119 for more information about other parameters in this section.

Related information:

"Shutting down the Client daemon" on page 205
You can shut down the Client daemon for all connected servers, after all outstanding units of work have completed or without completing outstanding units of work, and shut down the session with a particular server.

Send TCP/IP KeepAlive packets:

The **tcpkeepalive** parameter specifies if you want TCP/IP to periodically send keepalive messages to the server to check the connection.

Select **Send TCP/IP KeepAlive packets** in the Configuration tool, if you want TCP/IP to periodically send keepalive messages to the server to check the connection.

Set to **Y** to enable. This setting is disabled by default.

The CICS Transaction Gateway uses the interval specified by your operating system.

This parameter is in the SERVER section of the configuration file, see "SERVER section of the configuration file" on page 119 for more information about other parameters in this section.

Use uppercase security:

The **uppercasesecurity** parameter specifies that the CICS Transaction Gateway converts to uppercase any user ID or password from an ECI application or from a user prompt.

Select **Use uppercase security** in the Configuration tool to specify that the CICS Transaction Gateway converts to uppercase any user ID or password from an ECI application or from a user prompt.

Set the value of **uppercasesecurity** to **N** in the configuration file to disable upper casing of user IDs and passwords.

The default is for the setting is **Y**.

This parameter is in the SERVER section of the configuration file, see "SERVER section of the configuration file" on page 119 for more information about other parameters in this section.

Configuring SNA

Perform these steps to configure an SNA server connection.

You can choose from two principal topologies:

- Full Communications Server installed on the local computer (full SNA stack).
- Communications Server Remote API Client (split SNA stack), supported only for Linux and AIX Communication Servers.

You need to install and configure an SNA communications server, such as IBM Communications Server. To set up communication over SNA define the following in your SNA communications product.

- The **local node characteristics** that are common to all SNA users at the workstation.
- A **local logical unit (LU)** for the CICS Transaction Gateway.
- A **partner logical unit** for each CICS server with which the CICS Transaction Gateway will communicate. This is not required if you are using APPN and specify the actual fully-qualified partner LU names in the CICS TG configuration file.
- One or more **modes** to specify sets of session properties that are used in binding SNA sessions.
- A **transaction program (TP)** for the CRSR transaction. You need this if the following apply:
  - The CICS servers support terminal emulation, and
  - You require automatic transaction initiation (ATI) against the CICS Transaction Gateway terminals.

The terms used to describe these definitions vary with the product used to provide support. The terms used above are the ones used by IBM Communications Server.

SNA links to the CICS Transaction Gateway support data synchronization levels (sync levels) 0 and 1.

Overview of SNA configuration definitions

To configure an SNA connection, you must make entries on VTAM, CICS, Communications Server, and the CICS Transaction Gateway.

The following table shows the entries that you need to make.

Table 3. Matching definitions for SNA

VTAM	CICS Transaction Server	SNA Server	ctg.ini	Example
NETID	—	First part of fully qualified LU name in Partner LU	—	ABC3XYZ4
PU	—	Control Point alias in Node Definition	—	IYAMR021
LU	Netname	LU Name/LU alias in independent LU Type 6.2	Local LU name	IYAMT210
XID	—	Last five digits of Node identifier in Node Definition	—	05d**316fc**
Token Ring destination address	—	Adjacent node MAC address in Link Station	—	400045121088
Ethernet port address	—	MAC address		020070000428
Enterprise Extender IP address	—	Communication end point		192.113.36.200
APPL	APPLID	Second part of fully qualified LU name in Partner LU	—	IYCQST34
LogMode	Modename	Name in Mode	Mode name	LU62PS
—	—	—	Partner LU name	IYCQST34

Note:

1. The NETID is the VTAM network name. It is defined for your VTAM network in the VTAM start options.
2. The PU is the name of the Communications Server physical unit (PU). It is named in the VTAM switched major node.
3. The LU is the independent LU6.2 used by the CICS Transaction Gateway. It must also be defined to VTAM in a switched Major Node.
4. The XID (or node ID) is configured in the VTAM switched major node using IDBLK and IDNUM. It is used in the XID exchange to activate the link station.

5. For IBM Communications Server, Token Ring, Ethernet and Enterprise Extender are all valid communication link station types. Choose one or more as required. For more details about link station configuration refer to the Communications Server Task Guides.
6. The APPL is the CICS APPLID and also the VTAM APPL. It is defined in the VTAM application major node used by the CICS server.
7. The LogMode is the mode group used to control LU6.2 session properties. It must be defined in a VTAM logon mode table, which must be named on the VTAM APPL definition.
8. The Host system control point name is the CP for the PU of the VTAM front end processor.

More information on connecting the CICS Transaction Gateway to CICS Transaction Server for z/OS is in *CICS Transaction Gateway V5 - The WebSphere Connector for CICS, SG24-6133*.

Configuring IBM Communications Server for Linux

Communications Server for Linux requires a number of environment variables to be defined.

Refer to the README file supplied with the SNA product and ensure that all variables are correctly set, including **LD_PRELOAD**. It is recommended that you set **LD_PRELOAD** in ctgd.conf so that the library is only loaded for the Client daemon process. The **LD_PRELOAD** environment variable must be set to a 32-bit shared object to allow the 32-bit Client daemon process to communicate with the SNA product. Read the man page for `ld.so` for information about security issues associated with LD variables.

To ensure that messages can be written to error logs, the user who starts the Client daemon process must be a member of the "sna" group.

Using IBM Communications Server for Linux Remote API client V6.3

IBM Communications Server for Linux Remote API Client V6.3 and later does not require the **LD_PRELOAD** environment variable of the Linux streams (LiS) library. To stop the Client daemon generating warning CCL4678W, set **LD_PRELOAD** to an empty string, for example:

```
export LD_PRELOAD=""
```

Configuring for ATI

To enable ATI against Communications Server client terminals, define the transaction program CRSR on the Server.

To define the transaction program using X-Window System:

1. Start the administration application **xsnapadmin** on HP-UX or **xsnaadmin** on all other platforms.
2. Select **Services**—>**APPC**—>**Transaction Programs**.
3. Select **TP invocation**.
4. Click **Add** or, on Linux, **New**.
5. Enter `CRSR` as the Application TP.
6. Indicate Parameters are for invocation on any LU Queue incoming allocates and enter the path to the executable file as: `<install_path>/bin/cclclnt`
7. Set **Arguments** to `CRSR`

8. Set **Userid** and **Group** to the user ID and group that the application will run under.
9. Click **OK** to close the TP definition window and save your parameters.
10. Click on **Local LU** > **Properties** > **Advanced** and ensure that the Attach routing computer host name is entered in the **Local LU advanced** parameter. The Attach routing computer name option is only available when the Communications Server is running as part of a client/server domain.
11. If you are using a split SNA stack, on the remote API client ensure that invoked_tps = YES is set in the sna_clnt.net configuration file.

Defining SNA connections on CICS Transaction Server for z/OS

This topic provides an overview of how to define SNA connections on CICS.

To define SNA connections on CICS do the following:

1. Specify the SIT parameter ISC=YES.
2. Install CSD groups DFHCLNT and DFHISC.
3. Create and install CICS connection and sessions definitions. For more information see "Defining CICS sessions."

For information about configuring SNA on other CICS servers, refer to your server documentation.

Defining the location of the remote system:

Define the location of the remote CICS system and the parameters of the connection to it, if you are not using autoinstalled connections.

Use CEDA to create a CONNECTION definition with the following settings:

ACcessmsmethod
Set to `Vtam`.

PRotocol
Set to `Appc`.

Singlesess
Set to `No`.

AUtoconnect
Specify whether CICS is to bind sessions (drive CNOS) when the connection is installed. Set this to `Yes`.

ATtachsec
This defines the settings for SNA LU6.2 conversation-level security. Set it to one of the following:

- `Verify` to flow a user ID and password from the CICS Transaction Gateway.
- `Local` if you do not want to flow a user ID and password.

Netname
Set to the LU name of the CICS TG, this is the LOCALLUNAME parameter specified in the SERVER definition in the CICS TG configuration file.

If you change and reinstall the CICS connection definition, you must stop and restart the connection.

Defining CICS sessions:

For each CICS connection definition, define one or more session definitions to specify the SNA mode groups to be used within that connection, if you are not using autoinstalled connections.

Use CEDA to create a SESSION definition with the following settings:

Connection
Set to the name of the associated connection definition.

MOdename
Specify the mode group as defined in a VTAM® LOGMODE. The modename must be unique among the sessions definitions that relate to one connection definition.

Protocol
Set to `APPC`.

MAximum
Specify the maximum number of sessions that are to be supported. The first value is the maximum number of sessions that can be supported. The second value specifies the number of contention-winner sessions (CICS-owned sessions). These values are negotiated during change number of sessions (CNOS) flows, when the sessions are actually bound; the negotiated values depend on the settings specified in the partner SNA node.

Set the first value to be at least as big as the MAXREQUESTS parameter in the ctg.ini file, to prevent a bottleneck to throughput. Set the second value to `001`, to ensure that START requests are shipped serially from the server to the client.

Autoconnect
Set to determine whether the sessions for this mode group will be bound when the connection is installed. Set it to one of the following:

- `Yes` to bind only contention-winner sessions
- `All` to bind all sessions

Configuring CICS connection autoinstall:

Autoinstall of connections is particularly useful when dealing with many similar connections or when you are unsure of the LU names (netnames) to be used.

To configure autoinstall of connection definitions:

1. Update the default autoinstall program from DFHZATDX to DFHZATDY, by specifying the SIT parameter AIEXIT=DFHZATDY. Alternatively, write your own autoinstall user-replaceable module based on the samples provided.
2. Configure model definitions. The supplied DFHZATDY autoinstall program uses the template CBPS. CBPS is supplied in DFHAI62 group; copy it to your own group and modify it accordingly. The parameters in the connection definition template are the same as for a static definition (see “Defining the location of the remote system” on page 64), except that the netname is not needed.

   The parameters for the sessions definition are the same as those listed in “Defining CICS sessions” on page 64, except that the Connection parameter must refer to the CBPS connection definition.

If you use the supplied connection autoinstall program (DFHZATDY), the connection name generated is based on the last four characters of the Netname. To change the connection name, create your own user-replaceable module from the sample provided in CICSTSxx.CICS.SDFHSAMP, where xx is the release of your

CICS TS server. For example, xx is replaced by 22 for CICS TS 2.2 (CICSTS22.CICS.SDFHSAMP), 23 for CICS TS 2.3, 31 for CICS TS 3.1, and 32 for CICS TS 3.2.

Configuring an SNA CICS Server definition

Use the CICS Transaction Gateway configuration tool to configure a new SNA CICS server definition, or edit the SERVER section of the configuration file directly.

To configure a new SNA CICS Server definition using the Configuration Tool:

1. Select **New Server** from the **Options** menu, or from the toolbar, or right click on the CICS Servers entry in the Navigation Panel.
2. Select a server in the navigation panel to display the Server connection panel. The settings map to the parameters in a Server section of the configuration file.

   Multiple identical server definitions are not permitted in the configuration file of the CICS Transaction Gateway. The combination of fields that identify an SNA server are Partner LU name, Local LU name and Mode name.

   This ensures that every connection that the CICS server and the network protocol see is represented by a unique server definition in the configuration file. If you have existing Client applications that use different server names to send requests to the same CICS server, you need to write a user exit to redirect requests. This is demonstrated in sample user exits `ecix2.c` and `epix2.c`; for more information, see the *CICS Transaction Gateway for Multiplatforms: Application Programming GuideCICS Transaction Gateway for z/OS: Application Programming Guide*.
3. Set the values required as described in the following sections.

To configure an SNA server definition by editing the configuration file directly, see "SERVER section of the configuration file" on page 119.

Server name:

The **`SECTION SERVER`** parameter provides a server name that is independent of the communications protocol for the server, local to the CICS Transaction Gateway.

Set the value in the range of 1 to 8 characters in the **Server name** field of the Configuration tool.

Supported characters are in the ranges A-Z and 0-9, and '@', '#', '$', '-'. Lower case characters you enter in the range a-z are converted to upper case. If you do not use single-byte characters, other characters might not be displayed correctly.

Use this name for all requests to access the server from Client applications.

This parameter is in the SERVER section of the configuration file, see "SERVER section of the configuration file" on page 119 for more information about other parameters in this section.

Description:

The **`description`** parameter specifies a description for the server.

Set the value in the range between 1 and 60 characters in the **Description** field of the Configuration Tool, in single-byte characters. Use single-byte characters to avoid other characters being displayed incorrectly. The value is optional.

The description is returned on list systems calls.

Depending on the configuration used, the value is defined in one of the following locations:

- "IPICSERVER section of the configuration file" on page 119
- "SERVER section of the configuration file" on page 119

Network protocol:

The **protocol** parameter specifies the protocol to be used for the server connection.

Select the protocol to be used for the server connection. This can be on of the following values:

- TCP/IP
- SNA
- IPIC

You must include a SECTION DRIVER for the protocols listed above, excluding IPIC, for example, SECTION DRIVER = SNA, and specify the correct **drivername** parameter. See "DRIVER section of the configuration file" on page 120 for more information.

If you use configuration tool, select the protocol to be used from the drop down list in the in the **Network protocol** field.

Initial transaction:

The **initialtransid** parameter specifies a transaction identifier of between 1 and 128 characters.

This string is case-sensitive and identifies the initial transaction, and any parameters, to be run when the terminal emulator connects to the server. If you do not enter anything in the **Initial transaction** field of the Configuration tool, no initial transaction is run. The first four characters, or the characters before the first blank in the string are taken as the transaction. The remaining data is passed to the transaction on its invocation.

Ensure that the transaction does not require terminal input.

This parameter is in the SERVER section of the configuration file, see "SERVER section of the configuration file" on page 119 for more information about other parameters in this section.

Model terminal definition:

The **modelterm** parameter specifies the name of a model terminal definition at the server.

Set the value of the string between 1 and 16 characters in the **Model terminal definition** field of the Configuration tool. The string is case-sensitive and identifies the characteristics of terminals to be autoinstalled from the client. If the model cannot be located at the server, or you do not enter anything, a default terminal definition is used. This default is server-specific.

The interpretation of the Model terminal definition setting is server-specific. For example, for a TXSeries for AIX server, the value is 1 to 16 characters, and is the DevType for a CICS terminal definition entry to be used as the model.

This parameter is in the SERVER section of the configuration file, see "SERVER section of the configuration file" on page 119 for more information about other parameters in this section.

Local LU name:

The **locallname** parameter specifies the local LU name alias as it is known to the SNA Communications Server.

Set the value in the **Local LU name alias** field of the Configuration tool. The value must be 8 characters or less and is used when connecting to the server.

For the Client daemon: this parameter is in the SERVER section of the configuration file. For information about the other parameters in this section, see "SERVER section of the configuration file" on page 119.

Partner LU name:

The **netname** parameter specifies the LU name of the CICS server as it is defined for the SNA network.

- If you selected **Use Partner LU alias name**, enter an alias name of eight characters or less in the **Partner LU name** field of the Configuration tool.
- If you did not select **Use Partner LU alias name**, enter a qualified name of up to 17 characters in the **Partner LU name** field of the Configuration tool ,for example, `ABC3XYZ4.PQRS1234`

This parameter is in the SERVER section of the configuration file, see "SERVER section of the configuration file" on page 119 for more information about other parameters in this section.

Use Partner LU alias name:

The **partnerlualias** parameter specifies that the **Partner LU name** is an alias.

Select **Use Partner LU alias name** in the Configuration tool to specify that the **Partner LU name** is an alias.

Set the value to **Y** to enable.

On **UNIX and Linux operating systems:** The default is for this to be selected.

This parameter is in the SERVER section of the configuration file, see "SERVER section of the configuration file" on page 119 for more information about other parameters in this section.

Mode name:

The **modename** parameter specifies the mode name to be used when connecting to the server.

Set the value between 1 and 8 characters in the **Mode name** field of the Configuration tool. Enter `*` if you want the mode name to be filled with spaces.

This parameter is in the SERVER section of the configuration file, see "SERVER section of the configuration file" on page 119 for more information about other parameters in this section.

Server idle timeout:

The **srvidletimeout** parameter specifies in minutes, the period of inactivity after which the connection between the Client daemon and the CICS server is closed.

Set an integer in the range 1 through 1080 in the **Server idle timeout (min)** field of the Configuration tool, a value of 0 means that no timeout is applied. The default value is 0.

The field is available if one of the following network protocols is selected:

- TCP/IP
- SNA
- IPIC

The **Server idle timeout (min)** period is counted from when the number of outstanding conversations (units of work) on the connection is zero. The connection is automatically reestablished when a Client application sends the next ECI, EPI or ESI request.

When a server connection times out, the Client daemon behaves as if the `cicscli -x=<servername>` command had been issued. In particular, any value specified in the "Server retry interval (Client daemon connections to CICS)" on page 87 configuration field is ignored, and no attempt is made to reestablish the connection. The Server retry interval setting is re-enabled if the connection is re-established.

If the network protocol for the server in question is not supported for this field, the Client daemon ignores any entry in the configuration file for the **Server idle timeout (min)** field.

This parameter is in the SERVER section of the configuration file, see "SERVER section of the configuration file" on page 119 for more information about other parameters in this section.

Related information:

"Shutting down the Client daemon" on page 205
You can shut down the Client daemon for all connected servers, after all outstanding units of work have completed or without completing outstanding units of work, and shut down the session with a particular server.

Use uppercase security:

The **uppercasesecurity** parameter specifies that the CICS Transaction Gateway converts to uppercase any user ID or password from an ECI application or from a user prompt.

Select **Use uppercase security** in the Configuration tool to specify that the CICS Transaction Gateway converts to uppercase any user ID or password from an ECI application or from a user prompt.

Set the value of **uppercasesecurity** to **N** in the configuration file to disable upper casing of user IDs and passwords.

The default is for the setting is **Y**.

This parameter is in the SERVER section of the configuration file, see "SERVER section of the configuration file" on page 119 for more information about other parameters in this section.

Configuring Gateway daemon settings

The Gateway daemon settings are defined in the ctg.ini configuration file and are used for remote mode scenarios. The settings control the Gateway daemon and its protocol handlers for remote client connections.

Gateway daemon resources

Use the CICS Transaction Gateway configuration tool to configure the Gateway daemon resources, or edit the GATEWAY section of the configuration file directly.

Initial number of connection manager threads

The **initconnect** parameter controls the number of connection manager threads created on start up which are available for client connections. Set the value to the usual number of JavaGateway objects opened by all connected clients.

Set the value in the range 1 through 1,000,000, in the **Initial number of connection manager threads** field of the Configuration tool, to specify the initial number of connection manager threads. The default is 1. You might need to set this number to less than the supported maximum value because of constraints on memory or other system resources.

You can override this setting with the **ctgstart -initconnect=***number* command. See "Starting the Gateway daemon with override options" on page 193 for more information.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Maximum number of connection manager threads

The **maxconnect** parameter limits the maximum number of JavaGateway objects opened by all connected Client applications. Set it to the maximum number of JavaGateway objects that could be open at any one time from all the remotely connected Client applications.

Set the value in the range 1 through 1,000,000 in the **Maximum number of connection manager threads** field of the Configuration tool, to specify the maximum number of connection manager threads. The default is 100. You might need to set this number to less than the supported maximum value because of constraints on memory or other system resources.

If you select **Unrestricted**, in the Configuration tool, no limits are applied to the number of connection manager threads.

You can override this setting with the **ctgstart -maxconnect=***number* command.

For information on threading limits, see "Threading model" on page 176.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

CICS Transaction Gateway Desktop Edition: The maximum value for this parameter is 5.

Initial number of worker threads

The **initworker** parameter controls the number of worker threads created on start up, which are available for processing client requests. Set the value to the usual number of concurrent requests expected to be processed concurrently by the Gateway daemon.

Set the value in the range 1 through 1,000,000, in the **Initial number of worker threads** field of the Configuration tool, to specify the initial number of worker threads. The default is 1. You might need to set this number to less than the supported maximum value because of constraints on memory or other system resources.

You can override this setting with the **ctgstart -initworker=***number* command.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Maximum number of worker threads

The **maxworker** parameter limits the maximum number of parallel ECI, ESI, and EPI requests that CICS Transaction Gateway can process.

Set the value in the range 1 through 1,000,000, in the **Maximum number of worker threads** field of the Configuration tool, . The default is 100. You might need to set this parameter to a value less than the maximum because of limitations on memory or other system resources.

You can override this setting with the **ctgstart -maxworker=***number* command.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

CICS Transaction Gateway Desktop Edition: The maximum value for this parameter is 5.

Enable reading input from console

The **noinput** parameter enables the reading of input from the console.

The **Enable reading input from console** field of the Configuration tool is enabled by default.

See also the `-quiet` command line option ("Starting the Gateway daemon with preset options" on page 193).

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

ECI generic replies

The **ecigenericreplies** parameter allows Java Client applications to obtain generic ECI replies from the CICS Transaction Gateway. The default is that generic replies are not enabled; enabling generic replies is not supported.

Set **ecigenericreplies=off** to disallow ECI generic replies. The use of the call types ECI_GET_REPLY or ECI_GET_REPLY_WAIT by Java client applications to obtain generic ECI replies is no longer supported. Do not select **Let Java Clients obtain generic ECI replies** in the Configuration tool, because this allows generic replies.

Generic replies are those obtained using the Call_Type: ECI_GET_REPLY or ECI_GET_REPLY_WAIT. Specific replies are those obtained using the Call_Type: ECI_GET_SPECIFIC_REPLY or ECI_GET_SPECIFIC_REPLY_WAIT.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Validate Units of Work

Set the **uowvalidation** parameter to validate logical units of work (LUWs). The default is that LUW validation is enabled; disabling validation is not supported.

Do not clear **Validate Units of Work** in the Configuration tool because this disables validation.

Set **uowvalidation=on** to validate LUWs, this ensures that an LUW ID can be used only on the JavaGateway connection to which it was allocated.

If you disable validation, the following are true:

- LUWs can be accessed by any connection to the same remote Gateway.
- The CICS Transaction Gateway cannot clean up used LUWs when a connection is closed or breaks.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Validate message qualifiers

The **msgqualvalidation** parameter specifies if you want the CICS Transaction Gateway to validate message qualifiers. The default is that message qualifier validation is enabled; disabling validation is not supported.

This ensures that a message qualifier ID can be used only on the JavaGateway connection to which it was allocated. A message qualifier that has been assigned on an asynchronous call cannot be used by any connection using the same remote Gateway until the reply has been received. Set **msgqualvalidation=on** to enable validation. Do not clear **Validate message qualifiers** in the Configuration tool because this disables validation.

If you disable validation the following are true:

- ECI asynchronous calls tagged with a message qualifier on one connection can have a call of the GET_SPECIFIC_REPLY type made from another connection to the same remote gateway.

- A message qualifier can be used on an asynchronous call even if a reply with the same message qualifier is still outstanding in the remote Gateway.
- The CICS Transaction Gateway cannot clean up used message qualifiers when a connection is closed or breaks. In some circumstances logical units of work (LUWs) will also not be cleaned up: because the Gateway cannot clean up the message qualifier, it cannot determine if the LUW is still active.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Timeout for in-progress requests to complete

The **closetimeout** parameter specifies the timeout for in-progress requests to complete in milliseconds.

Set the value in the range 0 through 1,000,000, in the **Timeout for in-progress requests to complete** field of the Configuration tool, to specify the value in milliseconds. The default timeout is 10,000 milliseconds.

When a Java Client application disconnects from the CICS Transaction Gateway, the Gateway might still be processing requests on behalf of that program, for one of the following reasons.

- The connection manager thread that was managing requests on behalf of the Java Client application waits for outstanding requests to complete for up to the timeout period. If this field is set to zero, the connection manager thread moves immediately to the following step.
- After the timeout has expired, the connection manager thread closes the protocol handler and returns any worker threads without in-progress requests to the pool.
- When all in-progress requests have completed, the connection manager thread returns itself to the pool for reuse.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Worker thread availability timeout

The **workertimeout** parameter specifies the timeout period for a worker thread to become available, in milliseconds. When a connection manager thread accepts a request, it must allocate a worker thread to run that request. If a worker thread does not become available within the timeout period, an error message is sent rejecting that request and the request is not run.

Set the **workertimeout** parameter in the range 0 through 1,000,000, in the **Worker thread available timeout (ms)** field of the Configuration tool, to specify the value in milliseconds.

The default timeout is set to 10,000 milliseconds, but you can set a value to override that default.

If you set this value to zero, the request is rejected unless a worker thread is immediately available.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Port for local administration

The **adminport** parameter specifies the port for local administration.

Set the value in the range 1 through 65,535, in the **Port for local administration** field of the Configuration tool, to specify the port number on which to listen for administration requests. The default is 2810.

You can override this setting with the **ctgstart -adminport=***number* command.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Maximum number of connections

The **maxconn** parameter specifies the maximum number of applications that can be connected at the same time to perform statistic queries.

Set the value in the **Maximum number of connections** field of the Configuration tool, to the maximum number of connections. The default is 5.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Gateway daemon logging

Use the CICS Transaction Gateway configuration tool to configure the Gateway daemon logging resources, or edit the GATEWAY section of the configuration file directly.

Log destinations

The **log@error.dest** parameter specifies the destination for error and warning messages displayed by the Gateway daemon. The **log@info.dest** parameter specifies the destination for informational messages displayed by the Gateway daemon.

Specify the destination for error and warning messages in the **Error and warning log destination** field of the Configuration tool. Specify the destination for information messages in the **Information log destination** field of the Configuration tool.

The File destination must be used when the Gateway daemon is run as a background process. The *Console* option sends output to stderr.

These parameters are in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Log file names

The **log@error.parameters filename** parameter specifies the name of the error and warning log file to be used for problem diagnosis. The **log@info.parameters filename** parameter specifies the name of the information log file to be used for information messages.

By default, information messages are logged to the error and warning log file.

The file name cannot contain the percent sign (%) character. Use either a forward slash (/) character or double backslash (\\) characters as a separator in the path name on all platforms.

Specify the name of the log file to be used for error and warning messages in the **Error log file name** field of the Configuration tool. Specify the name of the log file to be used for information messages in the **Information log file name** field of the Configuration tool.

For example:

```
/var/logs/cicstg.log
```

The directory specified in the filename must exist before the Gateway daemon is started.

When the filename parameter is defined without a directory the log file is created in the <install_path>/bin directory.

See "Location of product files" on page 21 for more information.

These parameters are in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Maximum file sizes

The **log@error.parameters filesize** parameter specifies the maximum size in kilobytes of the error log file. The **log@info.parameters filesize** parameter specifies the maximum size in kilobytes of the information log file.

Set the value in the range 0 through 2,097,151 in the **Error log maximum size (KB)** field of the Configuration tool and the **Information log maximum size (KB)** field of the Configuration tool. A value of 0 means that no limit is placed on the file size. The default value is 0. When **filesize** is greater than 0 **maxfiles** must be changed to be greater than 1.

If you set the maximum size field for a log, you must also set the number of archived logs to keep field greater than 1. Do not set a maximum file size and set the number of archived logs to 1. This results in the issue of error message CTG8413E and all messages are redirected to the console. No log file is created..

These parameters are in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Maximum number of files

The **log@error.parameters maxfiles** parameter specifies the maximum number of error log files that are maintained. The **log@info.parameters maxfiles** parameter specifies the maximum number of information log files that are maintained.

Set the value in the range 1 through 9999, in the **Maximum number of error log files** and **Maximum number of information log files** fields of the Configuration tool. The default value is 1. Any entry in this field is ignored if the "Maximum file sizes" field has a value of 0.

When **Error log maximum size (KB)** or **Information log maximum size (KB)** is greater than 0 you must set **Maximum number of error log files** or **Maximum number of information log files** to be greater than 1. A value greater than 1 results in the file name of the log file being suffixed by sequence numbers until the number of files specified in maxfiles have been created. For example, if you set "Log file names" on page 74 to *ctg.log*, and the **Maximum number of error log files or Maximum number of information log files** field to 4, at most the following files will be created:

```
ctg.log.3  This is the oldest log file.
ctg.log.2
ctg.log.1
ctg.log.0  This is the log file currently being written to.
```

In other words, the smaller the filename suffix the newer the log file.

These parameters are in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Log Client connections and disconnections

The **connectionlogging** parameter controls whether or not the CICS Transaction Gateway writes a message to the log each time that a Client application connects to, or disconnects from, the Gateway daemon.

Set the value to **on** to enable this option. If you are using the configuration tool, select the **Log Client connections and disconnections** check box. The default is for these messages not to be written.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Log CICS messages

The **cicslogging** parameter controls whether or not messages returned from CICS in IPIC error flows are logged to the CICS TG error log.

Set the value to **on** to enable logging of messages returned from CICS. The messages are logged within a CICS Transaction Gateway warning message. If you are using the Configuration Tool select the **Log messages received from CICS** check box. The default is for these messages not to be written.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Display TCP/IP hostnames

The **dnsnames** parameter allow you to choose how TCP/IP addresses are displayed in messages.

By default, CICS Transaction Gateway displays TCP/IP addresses in messages in numeric form. If you enable this option, CICS Transaction Gateway uses the Domain Name System (DNS) to convert numeric TCP/IP addresses to symbolic TCP/IP host names in messages. This conversion makes the messages easier to read but might cause a significant reduction in performance.

Set the **dnsnames** parameter to **on** to enable this option. If you are using the Configuration tool select the **Display TCP/IP hostnames** check box.

Note: The **dnsnames** parameter supersedes the **nonames** parameter.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Timing information

The **notime** parameter disables timing information in messages.

Set the **Timing information** field of the Configuration tool to **on** to disable timing information messages.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Console output

The **quiet** parameter suppresses all console output.

Set the **Console output** field of the Configuration tool to **on** to suppress console output.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

TCP protocol settings

Use the CICS Transaction Gateway configuration tool to configure the TCP protocol settings, or edit the TCP protocol parameters in the GATEWAY section of the configuration file directly.

Bind address

The **bind** parameter sets the IP address or name of the host to which the protocol handler is to be bound.

If you set a host name in the **Bind address** field of the Configuration tool, it is resolved on startup. If the bind parameter is not specified or is blank, the default behavior is to bind to all IP addresses.

The IP address can be in IPv6 format; for example, `3ffe:307:8:0:260:97ff:fe40:efab`.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Port

The **port** parameter specifies the number of the TCP/IP port on which the protocol handler listens for incoming client requests.

Set the value in the range 1 through 65,535 in the **Port** field of the Configuration tool.

The default port for TCP/IP is 2006, for SSL the default is 8050, and for the statistics API protocol the default is 2980.

You can override the port setting as follows:

- For TCP/IP use the **ctgstart -port=***number* command
- For SSL use the **ctgstart -sslport=***number* command
- For the statistics API use the **ctgstart -statsport=***number* command

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Connection timeout (ms)

The **connecttimeout** parameter specifies how long the protocol handler waits for a connection manager thread to become available.

Set the value in the range 0 through 65,536 in the **Connection timeout (ms)** field of the Configuration tool, to specify the value in milliseconds. The default is 2000.

When a new connection has been accepted, the protocol handler waits for a connection manager thread to become available. If a connection manager thread does not become available within this time, the connection is refused. If this value is set to zero, a connection is refused if a connection manager thread is not immediately available.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Idle timeout (ms)

The **idletimeout** parameter specifies in milliseconds how long a connection is allowed to remain idle.

Set the value in milliseconds between 0 and 9,999,999 in the **Idle timeout (ms)** field of the Configuration tool.

The idle timeout period is counted from the time when a request was last flowed down the connection. When the idle timeout has expired, the client application is disconnected, although, if work is still in progress on behalf of the connection, the client application might remain connected, depending on the setting of the "Drop working connections" on page 79 parameter. If the idle timeout parameter is not set or is set to zero, idle connections are not disconnected.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Ping frequency interval (ms)

The **pingfrequency** parameter specifies how often a ping message is sent by the Gateway to an attached client to check that client is still active.

Set the value in the range 0 through 65,536 in the **Ping frequency interval (ms)** field of the Configuration tool, to specify the value in milliseconds.

If a reply has not been received by the time the next ping message is due to be sent, the connection is disconnected. Again, if work is still in progress on behalf of the connection it might, depending on the **dropworking** parameter value setting, be left connected. If this value is not set, or is set to zero, ping messages are not sent.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Drop working connections

The **dropworking** parameter specifies that a connection can be disconnected due to an idle timeout or a PING/PONG failure, even if work is still in progress on behalf of this connection.

Check **Drop working connections** in the Configuration tool to specify that a connection can be disconnected due to an idle timeout or a PING/PONG failure, even if work is still in progress on behalf of this connection.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

SO_LINGER setting

The **solinger** parameter sets the delay value in seconds for closing a socket.

Set the value in the range 0 through 65,536 in the **SO_LINGER setting** field of the Configuration tool, to specify the **SO_LINGER** setting for any socket used by this handler. If this value is not entered or is set to zero, **SO_LINGER** is disabled for any sockets used by this protocol handler.

If **SO_LINGER** is enabled, and data transmission has not finished, a call to close the socket blocks the calling program until the data is transmitted or until the connection times out. If **SO_LINGER** is disabled, a call to close the socket returns without blocking the caller and TCP/IP still tries to send the data. Normally, this transfer is successful, but it cannot be guaranteed, because TCP/IP repeats the Send request for only a specified period of time.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Require Java Clients to use security classes

The **requiresecurity** parameter allows your Gateway to accept only connections that use security classes.

Check **Require Java Clients to use security classes** in the Configuration tool to allow your Gateway to accept only connections that use security classes.

When a Java Client application connects to the Gateway, it can specify a pair of security classes for use on the connection. However, by default, a Gateway also accepts connections from programs that do not specify this pair of security classes.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

SSL protocol settings

Use the CICS Transaction Gateway configuration tool to configure the SSL protocol settings, or edit the SSL protocol parameters in the GATEWAY section of the configuration file directly.

Bind address

The **bind** parameter sets the IP address or name of the host to which the protocol handler is to be bound.

If you set a host name in the **Bind address** field of the Configuration tool, it is resolved on startup. If the bind parameter is not specified or is blank, the default behavior is to bind to all IP addresses.

The IP address can be in IPv6 format; for example, `3ffe:307:8:0:260:97ff:fe40:efab`.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Port

The **port** parameter specifies the number of the TCP/IP port on which the protocol handler listens for incoming client requests.

Set the value in the range 1 through 65,535 in the **Port** field of the Configuration tool.

The default port for TCP/IP is 2006, for SSL the default is 8050, and for the statistics API protocol the default is 2980.

You can override the port setting as follows:

- For TCP/IP use the **ctgstart -port=***number* command
- For SSL use the **ctgstart -sslport=***number* command
- For the statistics API use the **ctgstart -statsport=***number* command

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Connection timeout (ms)

The **connecttimeout** parameter specifies how long the protocol handler waits for a connection manager thread to become available.

Set the value in the range 0 through 65,536 in the **Connection timeout (ms)** field of the Configuration tool, to specify the value in milliseconds. The default is 2000.

When a new connection has been accepted, the protocol handler waits for a connection manager thread to become available. If a connection manager thread does not become available within this time, the connection is refused. If this value is set to zero, a connection is refused if a connection manager thread is not immediately available.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Idle timeout (ms)

The **idletimeout** parameter specifies in milliseconds how long a connection is allowed to remain idle.

Set the value in milliseconds between 0 and 9,999,999 in the **Idle timeout (ms)** field of the Configuration tool.

The idle timeout period is counted from the time when a request was last flowed down the connection. When the idle timeout has expired, the client application is disconnected, although, if work is still in progress on behalf of the connection, the client application might remain connected, depending on the setting of the "Drop working connections" on page 79 parameter. If the idle timeout parameter is not set or is set to zero, idle connections are not disconnected.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Ping frequency interval (ms)

The **pingfrequency** parameter specifies how often a ping message is sent by the Gateway to an attached client to check that client is still active.

Set the value in the range 0 through 65,536 in the **Ping frequency interval (ms)** field of the Configuration tool, to specify the value in milliseconds.

If a reply has not been received by the time the next ping message is due to be sent, the connection is disconnected. Again, if work is still in progress on behalf of the connection it might, depending on the **dropworking** parameter value setting, be left connected. If this value is not set, or is set to zero, ping messages are not sent.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Drop working connections

The **dropworking** parameter specifies that a connection can be disconnected due to an idle timeout or a PING/PONG failure, even if work is still in progress on behalf of this connection.

Check **Drop working connections** in the Configuration tool to specify that a connection can be disconnected due to an idle timeout or a PING/PONG failure, even if work is still in progress on behalf of this connection.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

SO_LINGER setting

The **solinger** parameter sets the delay value in seconds for closing a socket.

Set the value in the range 0 through 65,536 in the **SO_LINGER setting** field of the Configuration tool, to specify the **SO_LINGER** setting for any socket used by this handler. If this value is not entered or is set to zero, **SO_LINGER** is disabled for any sockets used by this protocol handler.

If **SO_LINGER** is enabled, and data transmission has not finished, a call to close the socket blocks the calling program until the data is transmitted or until the connection times out. If **SO_LINGER** is disabled, a call to close the socket returns without blocking the caller and TCP/IP still tries to send the data. Normally, this

transfer is successful, but it cannot be guaranteed, because TCP/IP repeats the Send request for only a specified period of time.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Require Java Clients to use security classes

The **requiresecurity** parameter allows your Gateway to accept only connections that use security classes.

Check **Require Java Clients to use security classes** in the Configuration tool to allow your Gateway to accept only connections that use security classes.

When a Java Client application connects to the Gateway, it can specify a pair of security classes for use on the connection. However, by default, a Gateway also accepts connections from programs that do not specify this pair of security classes.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Use client authentication

The **clientauth** parameter enables client authentication.

Check **Use client authentication** in the Configuration tool to use client authentication.

The default is for client authentication to be disabled.

When client authentication is enabled, any connection attempted to the ssl: handler requires the client to present its own Client Certificate (also known as a digital ID).

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Key ring file

The **keyring** parameter specifies the key ring server key ring name.

Set the value in the **Key ring file** field of the Configuration tool, as either the full path name, or the path name of the file relative to the CICS Transaction Gateway `bin` directory. Use either a forward slash (`/`) character or double backslash (`\\`) characters as a separator in the path name on all operating systems.

For example:

```
/mykeys/jsse/keystore.jks
\\mykeys\\jsse\\keystore.jks
```

The server key ring consists of a valid x.509 certificate that identifies this server to connecting clients. This key ring is generated using the SSL tools supplied with this product.

You can override the SSL key ring path and file name with the **ctgstart -keyring=***keyring* command.

If you are using a RACF keyring, the keyring parameter value is the name of the RACF key ring.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Key ring password

The **keyringpw** parameter value is the password that you specified for the server key ring. The **keyringpwscrambled** parameter specifies whether the **keyringpw** parameter value is encrypted. The default is off.

Set the value in the **Key ring password** field of the Configuration tool. The Configuration tool writes the password to the configuration file in a form that prevents an observer from easily reading it and sets **keyringpwscrambled** to on.

You can override the SSL password with the **ctgstart -keyring=***keyring* **-keyringpw=***keyringpw* command. An error message is generated if the keyringpw parameter is used on its own without the corresponding keyring parameter in the ctgstart - command line. This error message does not affect the Gateway operation and the password is ignored.

The following example shows a command line with parameters:

```
ctgstart -sslport=port_number -keyring=keyring -keyringpw=keyringpw
```

To encrypt this entry, ensure that the **keyringpwscrambled** parameter is set to on.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Use only these ciphers

Use the **ciphersuites** parameter to restrict the set of cipher suites that can be used with the SSL protocol.

If you do not want to restrict the set of cipher suites used, omit the **ciphersuites** parameter.

Click **Add** to add the cipher to the list of ciphers, below the entry field. Click **Remove** to remove all selected entries in the ciphers list. To restrict the cipher suites that can be used with the SSL protocol, enter a comma-separated list of cipher suites. To allow all available cipher suites to be used, omit the entry. CICS Transaction Gateway uses cipher suites provided by the Java runtime environment for the SSL protocol. The cipher suites available to be used are dependant on the Java version. See the documentation supplied with your Java runtime environment for valid cipher suites, or do the following:

1. Remove any entries from the **Use only these ciphers** list.
2. Save the configuration file.
3. Run ctgstart.

If the SSL protocol is correctly configured, CICS Transaction Gateway displays a list of valid cipher suites that Java client applications can use when connecting to the CICS Transaction Gateway.

If the **Use only these ciphers** list contains entries, a Java client application can connect to the CICS Transaction Gateway only by using the cipher suites listed. If the Java client application does not support any of the cipher suites listed, it cannot connect.

If the list contains no entries, a Java client application can connect using any available cipher suite.

Use of the **ciphersuites=128bitonly** parameter is deprecated.

If you use the Configuration Tool to open a configuration file that contains this entry, the entry is replaced by these cipher suites:

TLS_RSA_WITH_RC4_128_MD5

TLS_RSA_WITH_RC4_128_SHA

TLS_DHE_DSS_WITH_RC4_128_SHA

Cipher suites entered as TLS_ are converted to SSL_ when CICS Transaction Gateway starts. You can find the protocol used by checking the log or trace when a client connects.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Configuring Client daemon settings

Use the CICS Transaction Gateway configuration tool to configure the Client daemon settings, or edit the CLIENT section of the configuration file directly.

To display the Client daemon settings panel, select the Client daemon icon in the navigation panel tree structure.

Using the Configuration Tool, you can provide preset values for any parameter that can be specified using a Client command-line option.

If a parameter that has been defined using the Configuration Tool is subsequently specified by the associated command-line option when the Client is started, the command-line setting takes precedence.

Maximum buffer size

The **maxbuffersize** parameter specifies the size of the transmission buffers in which application data or terminal data flows.

Set the number of kilobytes value in the range 4 through 32 in the **Maximum buffer size** field of the Configuration tool. The default is 32 KB.

The value should be large enough to accept the largest possible COMMAREA or terminal input/output area (TIOA) to be used. The maximum COMMAREA size might be less than the Maximum buffer size, because certain protocols have an overhead of 512 bytes.

Leave this setting at the default unless the CICS Transaction Gateway is running on a machine that is short of memory.

This parameter is in the CLIENT section of the configuration file, see "CLIENT section of the configuration file" on page 118 for more information about other parameters in this section.

Terminal exit

The **terminalexit** parameter specifies a character string that when entered in place of a transaction name in a terminal emulator, causes the terminal emulator to terminate.

Set the value of the character string between 1 and 4 characters in the **Terminal exit** field of the Configuration tool. The default is EXIT.

The string, when entered at a terminal emulator at any time and place where a transaction name can be entered, causes the terminal emulator to terminate. The string must not contain any blank characters.

The string is case-sensitive. If a terminal emulator has uppercase translation in its CICS terminal definition, enter this string in uppercase.

This parameter is in the CLIENT section of the configuration file, see "CLIENT section of the configuration file" on page 118 for more information about other parameters in this section.

Maximum servers

The **maxservers** parameter specifies the maximum number of servers that can be accessed concurrently from the client.

Set the value in the range 1 through 256 in the **Maximum servers** field of the Configuration tool. The default is 10.

This parameter is in the CLIENT section of the configuration file, see "CLIENT section of the configuration file" on page 118 for more information about other parameters in this section.

Maximum requests

The **maxrequests** parameter specifies the maximum number of concurrent requests that might be executing on the Client daemon.

Set the value in the range 1 through 10,000 in the **Maximum requests** field of the Configuration tool. The default is 256.

An item is defined as one of the following:

- A request to install or uninstall a terminal emulator
- A request to install or uninstall an EPI terminal
- A transaction invoked by a terminal
- An ECI unit of work
- An ESI unit of work

The value is used to detect runaway conditions where an application could, in error, submit an excessive number of requests to a server. The actual limit might be less than this setting if other operating system limits (for example, memory constraint or communication sessions), come into effect.

This parameter is in the CLIENT section of the configuration file, see "CLIENT section of the configuration file" on page 118 for more information about other parameters in this section.

CICS Transaction Gateway Desktop Edition: The maximum value for this parameter is 32.

Print command

The **printcommand** parameter specified string is a command specific to the operating system under which the CICS Transaction Gateway is running. When a request to print is received, the Client daemon generates a temporary print file with a unique name.

Set the character string, from 1 to 256 characters long in the **Print command** field of the Configuration tool.

The parameter string is appended with the temporary file name, and the resultant command executed. This allows, for example, print requests to be copied to a file, directed to a local printer, formatted for inclusion into documentation, and other similar actions.

It is the responsibility of the Print command to delete the temporary print file after it has finished processing it.

Use a shell script with the Print command (for example, **lpr**) followed by the command to delete (**rm**).

See also the **Print file** description for more information.

This parameter is in the CLIENT section of the configuration file, see "CLIENT section of the configuration file" on page 118 for more information about other parameters in this section.

Print file

The **printfile** parameter identifies a file to which output from print requests received at the Client daemon is directed.

Set the file name to a character string of 1 to 256 characters long in the **Print file** field of the Configuration tool.

This option applies only if you make no entry in the **Print command** setting.

Each print request is appended to the end of the current file.

This setting acts only as a default. The **cicsterm** and **cicsprnt** commands provide options to override this value.

This parameter is in the CLIENT section of the configuration file, see "CLIENT section of the configuration file" on page 118 for more information about other parameters in this section.

Code page identifier override

The **CCSID** parameter indicates the Coded Character Set Identifier (CCSID) that identifies the coded graphic character set used by the client application for 3270 data flowed in EPI requests.

Use the **Codepage identifier override** field of the Configuration tool if your platform has been updated for euro support, and the CICS Server has euro support. For example, for Latin-1 countries, use a CCSID value of 858 to indicate that the code page 850 includes euro support. For code page 1252, specify a CCSID value of 5348.

Note:

1. Regardless of the value in the **Code page identifier override** setting, `cicsterm` always displays characters based on the local code page of the workstation.
2. If you use the CCSID to change the code page identifier, data that is already stored on the server might be modified when retrieved by the CICS Transaction Gateway, if it includes characters for which the code points produce different characters.
3. **AIX operating system:** On AIX 4.3.2 and above, support for Ja_JP locale uses CCSID=943. If the CICS Server does not support this CCSID, enter a code page identifier override of 932, or add the statement 'CCSID=932' to the Client Section of the configuration file ctg.ini.

This parameter is in the CLIENT section of the configuration file, see "CLIENT section of the configuration file" on page 118 for more information about other parameters in this section.

Server retry interval (Client daemon connections to CICS)

The **srvretryinterval** parameter specifies the time in seconds between attempts by the Client daemon to reconnect to a CICS server.

Set the value between 0 and 3600 in the **Server retry interval (sec)** field of the Configuration tool. The default is 60 seconds.

If the CICS server which is currently connected becomes inactive, an attempt is made to reconnect one second after the CICS server becomes inactive. If the connection attempt fails, additional attempts are made to connect at the interval specified by the **srvretryinterval** parameter.

Set the value to 0 to prevent automatic connection attempts. Each request directed at the CICS server initiates a connection attempt if one is not already in progress.

This parameter is located in the CLIENT section of the configuration file. For more information see "CLIENT section of the configuration file" on page 118.

Client daemon logging

You can change the file destination for log messages generated by the Client daemon.

Error, warning, and informational messages are output to the same log file by default, the log file is specified by the logfile parameter, see "Error and warning log file" on page 88. The destination of informational messages can be changed to be a separate file, see "Information log file" on page 88 for more information.

The language of the log messages can be changed by running the ctgmsgs command, see “Changing the system locale” on page 31 for more information. The log files can be viewed using any standard text editor. The following steps show how to change the settings.

1. Launch the Configuration Tool and navigate to the **Logging** tab of the **Client daemon** node.
2. Complete the fields on the tab as appropriate. You can choose to:
   - Log terminal installations and deletions.
   - Change the name of the Error and warning log file from the default cicscli.log.
   - Specify a different log for information messages.
3. Save the configuration file.

Related information:

“Client daemon logging” on page 87
You can change the file destination for log messages generated by the Client daemon.

Error and warning log file

The **logfile** parameter specifies the name of the log file to be used for error and warning messages.

Set the name of the logfile in the **Error log file name** field of the Configuration tool. If not specified, the log file name defaults to cicscli.log in the subdirectory /var/cicscli.

This parameter is in the CLIENT section of the configuration file, see “CLIENT section of the configuration file” on page 118 for more information about other parameters in this section.

Information log file

The **logfileinfo** parameter specifies the name of the file to which information messages will be logged.

By default, information messages are logged to the “Error and warning log file.”

To configure a separate information log file, clear **Use the same file for information messages** in the Configuration tool, and then complete the **Information log file** field.

If you do not specify a path to the file, the log is created in the following directory:

/var/cicscli

This parameter is in the CLIENT section of the configuration file, see “CLIENT section of the configuration file” on page 118 for more information about other parameters in this section.

Log terminal installations and deletions

The **terminstlogging** parameter specifies whether or not terminal installation and deletion requests are logged.

To activate logging of terminal installation and deletion requests, either select **Log terminal installations and deletions** in the Configuration Tool, or edit the configuration file to set the terminstlogging parameter to Y.

To deactivate logging, either deselect **Log terminal installations and deletions** in the Configuration Tool, or edit the configuration file to set the terminstlogging parameter to N.

Logging of terminal installation and deletion requests is deactivated by default.

The terminstlogging parameter is located in the CLIENT section of the configuration file. For information about the other parameters in this section see "CLIENT section of the configuration file" on page 118.

Configuring SSL

You can configure CICS Transaction Gateway to use the SSL cryptographic protocol for security and data integrity of communications over a TCP/IP connection.

If you are using IPIC connections, SSL is available in local mode only, using the resource adapter or base class API.

Creating and maintaining digital certificates

Digital certificates are used for identifying either end of a an SSL connection and contain information required to establish trust.

A digital certificate is a digitally signed data structure that binds a public key to the identity of the private key's owner. The use of digital certificates ensures that the user of a public key can be confident of the ownership of the corresponding private key. If you intend using SSL, you must always configure server authentication.

Server authentication tasks (mandatory for SSL)

1. Create a CA certificate on your Server which is self signed, or send a certificate request to an external CA and have it signed by them.
2. Generate a personal certificate on the Server and sign it with your CA certificate.
3. Export the personal certificate to a file on your Server.
4. Transfer the file to your Client.
5. Create a keystore/key ring on your Client and import the server personal certificate from the file into it.

Client authentication tasks (optional for SSL)

1. Create a CA certificate on your Client which is self signed, or send a certificate request to an external CA and have it signed by them.
2. Generate a personal certificate on the Client and sign it with your CA certificate.
3. Export the personal certificate to a file on your Client.
4. Transfer the file to your Server.
5. Import the Server personal certificate to the Client.

Tools for working with digital certificates

Use these tools to work with digital certificates in different scenarios:

The keytool utility is a command line tool; iKeyman is a graphical tool.

- keytool

- iKeyman

iKeyman and iKeytool are shipped in both the JRE and SDK packages.

Related information:

"Using keytool for certificate management" on page 93
The keytool command line application, provided with the SDK, is an accessible alternative to iKeyMan for managing self-signed certificates.

Configuring server authentication with iKeyman

You configure server authentication by creating a client keyring, importing the server's signer certificate, creating a server keyring and certificate, and exporting the server's signer certificate.

For information about configuring server authentication from the command line, see "Configuring your SSL server" on page 93.

Creating a server keyring

The key ring contains your server certificate, with its associated private key, and several signer certificates. SSL uses the certificate to identify the server to connecting clients.

1. Start iKeyMan.
2. Select **Key Database File —> New**.
3. From **Key Database Type**, select **JKS**.
4. In **File name** type a name for your key ring, such as *MyServerkey ring.jks*.
5. In **Location**, type a suitable location to store your server key ring.
6. Select **OK**.
7. Type a password for the key ring file.

   iKeyMan gives you an indication of the "strength" of your password. You might use a mixture of letters and numbers for your password which makes the password more resistant to "brute force" dictionary attacks.
8. Select **OK**.

The generated file MyServerkey ring.jks contains, by default, a selection of popular signer certificates as follows:

```
VeriSign Class 3 Public Primary Certificate Authority
VeriSign Class 2 Public Primary Certificate Authority
VeriSign Class 1 Public Primary Certificate Authority
RSA Secure Server Certificate Authority
Thawte Personal Basic CA
VeriSign Test CA Root Certificate
Thawte Personal Premium CA
Thawte Premium Server CA
Thawte Server CA
Thawte Personal Freemail CA
```

The server can verify clients with the VeriSign Class 1 through 3 Public Primary Certificate Authority signer certificates .

Creating a server certificate

Now you are ready to create the self-signed Server Certificate and store it along with its private key in your server key ring:

1. In iKeyMan, select **Create-> New Self-Signed Certificate**
2. Complete the certificate request. Some fields are optional, but you must fill in at least the following (examples are shown):

   Key Label
   exampleServerCert

   Version
   select *X509 V3*

   Key Size
   select *1024*

   Common Name
   This defaults to the name of the machine you are using

   Validity Period
   The default is 365 days
3. Select **OK**.

   iKeyMan generates a public/private key pair.
4. The self-signed Server Certificate appears in the Personal Certificates window. The certificate has the name you typed in the **Key Label** field, in this example *exampleServerCert*.
5. With *exampleServerCert* highlighted, select **View/Edit**.

   Notice that the information in the **issued to** (certificate requester) textbox is the same as that in the **issued by** (signer) textbox. To establish SSL connections with a server presenting this certificate, the client must trust the signer. To do this the client key repository must contain the signer certificate of the server presenting *exampleServerCert*.

Exporting the server's signer certificate

1. With *exampleServerCert* highlighted, select **Extract Certificate...**
2. In the **Data type** pull-down menu, select *Base64-encoded ASCII*.
3. Type the name and location of the text file containing your Server Certificate data. Our example uses *exampleServercert.arm*
4. Select **OK**.

Store the exported certificate in a safe place. Import it into any client repository that needs to communicate with this SSL server.

Creating a client keyring

A *client key ring* contains as a minimum, the signer certificate of the SSL server, and a client x.509 certificate, if client authentication is required. The process for creating a client key ring is similar to that for a server:

1. Start iKeyMan
2. Select **Key Database File —> New**
3. From **Key Database Type**, select **JKS**
4. In **File name** type a name for your key ring, such as *MyClientkey ring.jks*
5. In **Location**, type a suitable location to store your client key ring
6. Select **OK**
7. Type a password for the key ring file.
8. Select **OK**

Like the server key ring, the client key ring contains a default selection of popular signer certificates.

Importing the server's signer certificate

1. In iKeyMan select **Add**
2. Locate the stored Server Base64-encoded ASCII certificate file. In our example, this is *exampleServercert.arm*.
3. Select **OK**.
4. Give this signer certificate a unique label, for example, *My Self-Signed Server Authority*.
5. Select **OK**.

   This new signer certificate is added to the list of default signers.

Configuring client authentication with iKeyman

You configure client authentication by creating a client certificate and exporting the client's signer certificate.

For information about configuring client authentication from the command line, see "Configuring your SSL clients" on page 95.

Creating a client certificate

If the SSL handler used by the CICS Transaction Gateway is configured to support just server authentication, you do not have to create a client certificate as described here because the client key ring needs to contain just the signer certificate of the server, which you have just imported. You can use the generated MyClient key ring.jks file with CICS Transaction Gateway's SSL protocol, which is configured to support server authentication.

Client authentication requires the client key ring also to contain a self-signed Certificate that is used to identify the connecting client.

1. In iKeyMan, select **Personal Certificates** from the pull-down menu below the **Key database content** label.
2. Select **New Self-Signed...**
3. Complete the certificate request. Some fields are optional, but you must fill in at least the following (examples are shown):

   Key Label
   exampleClientCert

   Version
   Select *X509 V3*

   Key Size
   Select *1024*

   Common Name
   This defaults to the name of the machine you are using

   Organization
   The name of your organization

   Country
   Select a two character ID from the list

   Validity Period
   The default is 365 days

4. Select **OK**.

   iKeyMan generates a public/private key pair.
5. The self-signed Client Certificate appears in the Personal Certificates window. The certificate has the name you typed in the **Key Label** field, in this example *exampleClientCert*.

Exporting the client's signer certificate

1. With *exampleClientCert* highlighted, select **Extract Certificate...**
2. In the **Data type** pull-down, select *Base64-encoded ASCII*
3. Type the name and location of the text file containing your Server Certificate data. Our example uses *exampleClientcert.arm*
4. Select **OK**.

Store the exported certificate in a safe place. It must be imported into any server repository that needs to communicate with this SSL client.

Using keytool for certificate management

The keytool command line application, provided with the SDK, is an accessible alternative to iKeyMan for managing self-signed certificates.

In the production environment you might choose to use externally signed certificates, which are managed in a similar way.

Configuring your SSL server

To configure your SSL server you create a server key ring and certificate, export the server's signer certificate, and transfer the server certificate to the client.

Create a server key ring and server certificate

Issue the following command to create both the KeyStore and certificate:

```
keytool -genkey -alias aliasname -keysize numericvalue -dname distname
   -keystore location -keypass password -storepass password
   -keyalg algorithm
```

The options are:

-genkey
: Generates a key pair and wraps the public key into a self-signed certificate.

-alias *aliasname*
: Defines the alias name that identifies the store containing the self-signed certificate and private key.

-keysize *numericvalue*
: Defines the size of the key.

-dname *distname*
: Specifies the X.500 distinguished name to be associated with the alias. This is used as the issuer and subject fields of the self-signed certificate. The distinguished name consists of a number of fields separated by commas in the following format:

  Each *strvalue* is a string value. The meaning of the abbreviations is as follows:

  - cn = common name
  - o = organization

- ou = organization unit
- l = city/locality
- s = state/province
- c = country name

An example of an X.500 distinguished name is shown here:

```
"cn=someserver.location.ibm.com,o=IBM,ou=IBMGB,
   l=Winchester,s=Hants,c=GB"
```

Figure 6. An X.500 distinguished name

-keystore *location*
The key ring file location. For example: `ktserverss.jks`

-keypass *password*
The password used to protect the private key. Set this to the same value as the `-storepass` password, to enable the CICS Transaction Gateway to establish a connection over SSL.

-storepass *password*
The password used to protect the integrity of the key ring. Set this to the same value as the `-keypass` password, to enable the CICS Transaction Gateway to establish a connection over SSL.

-keyalg *algorithm*
The algorithm to be used to generate the key pair.

An example of this command is shown here:

```
keytool -genkey -alias exampleServerCert -keysize 1024
   -dname "cn=vmware2.location.ibm.com,o=IBM,ou=IBMGB,l=Winchester,s=Hants,c=GB"
   -keystore ktserverss.jks -keypass default -storepass default
   -keyalg RSA
```

Figure 7. Using the keytool command to create a key ring containing a single self-signed certificate

View the newly created certificate

Use a command similar to the following to view all certificates in the key ring, including the one you just created:

```
keytool -list -keystore storename -storepass password -v
```

Where the options are:

-list List the contents of the key ring.

-keystore *storename*
The name of the key ring containing the certificates you want to view.

-storepass *password*
The password needed to access the key ring.

-v Show details of the certificates in the key ring.

An example of the keytool command to view certificates is shown here:

```
keytool -list -keystore ktserverss.jks -storepass default -v
```

Figure 8. Using the keytool command to view certificates

Export the server's signer certificatd

The next step is to export the signer certificate and store it in a safe place. This can then be imported into the repository of any client that needs to connect to this SSL server.

The certificate is exported by using the following instance of the keytool command:

```
keytool -export -alias aliasname -keystore location
   -storepass password -file filename -rfc
```

Where the options are:

-export
Export a certificate.

-alias *aliasname*
Name of the key (in the key ring) to export.

-keystore *location*
The key ring location.

-storepass *password*
The password used to protect the integrity of the key ring.

-file *filename*
The name of the file to export the certificate to.

-rfc Export the certificate in RFC format (Base64 encoded ASCII).

An example of the keytool command to export a signer certificate is shown here:

```
keytool -export -alias exampleServerCert -keystore ktserverss.jks -storepass default
   -file exampleServerCertKT.arm -rfc
```

Figure 9. Using the keytool command to export the signer certificate

Transfer the server certificate to the client

If you use FTP to transfer the file, ensure that your FTP client is in binary mode.

Configuring your SSL clients

To configure your SSL clients you create a client key ring and import the server's signer certificate, create a self-signed certificate in the client. Next you export the client's signer certificate, and transfer the server certificate to the client. Finally you import the client signer certificate into the server's key ring file.

If your server does not use client authentication you complete the first task (create a client key ring and import the server's signer certificate) but you do not have to complete the other tasks.

Create a client key ring and import the server's signer certificate.

Issuing the following command to create the key ring and import the certificate:

```
keytool -import -alias aliasname -file certfile -keystore keystorefile
   -storepass password -noprompt
```

Where the options are:

-import
Import a certificate.

-alias *aliasname*
The name under which the certificate is to be stored.

-file *certfile*
The file that contains the certificate.

-keystore *keystorefile*
The key ring into which the certificate is to be imported.

-storepass *password*
The password used to protect the integrity of the key ring.

-noprompt
Removes the need to confirm that the certificate is imported.

An example of this command is shown here:

```
keytool -import -alias exampleServer -file exampleServerCertKT.arm -keystore clientStore.jks
   -storepass default -noprompt
```

Figure 10. Using the keytool command to create a key ring containing the server's signer certificate

Create a self-signed certificate in the client key ring

To create a new keystore containing a self-signed certificate use the following instance of the keytool command:

```
keytool -genkey -alias aliasname -keysize numericvalue -dname distname
   -keystore location -keypass password -storepass password
   -keyalg algorithm
```

The options are:

-genkey
Generates a key pair and wraps the public key into a self-signed certificate.

-alias *aliasname*
Defines the alias name that identifies the store containing the self-signed certificate and private key.

-keysize *numericvalue*
Defines the size of the key.

-dname *distname*
Specifies the X.500 distinguished name to be associated with the alias. This is used as the issuer and subject fields of the self-signed certificate. The distinguished name consists of a number of fields separated by commas in the following format:

Each *strvalue* is a string value. The meaning of the abbreviations is as follows:

- cn = common name
- o = organization
- ou = organization unit
- l = city/locality
- s = state/province
- c = country name

An example of an X.500 distinguished name is shown here:

```
"cn=someserver.location.ibm.com,o=IBM,ou=IBMGB,
   l=Winchester,s=Hants,c=GB"
```

Figure 11. An X.500 distinguished name

-keystore *location*
The key ring file location. For example: `ktserverss.jks`

-keypass *password*
The password used to protect the private key. Set this to the same value as the `-storepass` password, to enable the CICS Transaction Gateway to establish a connection over SSL.

-storepass *password*
The password used to protect the integrity of the key ring. Set this to the same value as the `-keypass` password, to enable the CICS Transaction Gateway to establish a connection over SSL.

-keyalg *algorithm*
The algorithm to be used to generate the key pair.

An example of the keytool command is shown here:

```
keytool -genkey -alias exampleClientCert -keysize 1024
   -dname "cn=John Doe,o=IBM,ou=IBMGB,l=Winchester,s=Hants,c=GB"
   -keystore clientStore.jks -keypass default -storepass default
   -keyalg RSA
```

Figure 12. Using the keytool command to create a key ring containing a single self-signed certificate

Export the client's signer certificate

This certificate must be imported into the keystores of all servers that the SSL client needs to connect to.

To export the certificate use the following instance of the keytool command:

```
keytool -export -alias aliasname -keystore location
   -storepass password -file filename -rfc
```

Where the options are:

-export
Export a certificate.

-alias *aliasname*
Name of the key (in the key ring) to export.

-keystore *location*
The key ring location.

-storepass *password*
The password used to protect the integrity of the key ring.

-file *filename*
The name of the file to export the certificate to.

-rfc Export the certificate in RFC format (Base64 encoded ASCII).

An example instance of the keytool command to export a signer certificate is shown here:

```
keytool -export -alias exampleClientCert -keystore clientStore.jks -storepass default
   -file exampleClientCertKT.arm -rfc
```

Figure 13. Using the keytool command to export the signer certificate

Transfer the server certificate to the client

If you use FTP to transfer the file, ensure that your FTP client is in binary mode. For details on importing the certificate, see step Create a client key ring and import the server's signer certificate.

Gateway daemon SSL configuration

To use SSL for connections between Java client applications and the Gateway daemon you must configure the SSL protocol in the configuration file, ctg.ini.

For more information see SSL protocol settings and Configuring SSL security between a Java Client and the Gateway daemon.

Using hardware cryptography

Cryptography within the existing JCE architecture gives Java 2 programmers security and performance advantages of hardware cryptography with minimal changes to existing Java applications.

To use hardware cryptographic function provided by the IBMJCECCA provider:

1. Edit the java.security file in the ${java-home}/lib/security directory so that it contains the following lines:

   ```
   security.provider.1=com.ibm.crypto.hdwrCCA.provider.IBMJCECCA
   security.provider.2=com.ibm.crypto.provider.IBMJCE
   ```

2. Copy the unrestricted policy files from the ${java-home}/demo/jce/policy-files/unrestricted directory to the ${java-home}/lib/security directory.

If you intend to use the keytool command to create JKS files that do not use hardware encryption:

1. Edit the java.security file to remove the line that references JCE4758.
2. Create the keystores.
3. If you intend to use hardware cryptography as well, reinstate the line in the java.security file.

Using the SSL protocol

To make a connection to CICS Transaction Gateway, the client application flows a request which specifies a URL.

Typically, the URL might be similar to this:

```
ssl://transGatewayMachine:8050
```

SSL configuration for IPIC connections

To enable SSL for IPIC connections you define a key ring and password, either for the Java base classes or for the resource adapter. SSL is supported only in local mode.

Configuring SSL for the Java base classes

To configure SSL for the Java base classes:

1. Create a java.util.Properties object
2. Add the following properties:
   a. **JavaGateway.SSL_KEYRING_CLASS, <keyring file location>**
   b. **JavaGateway.SSL_KEYRING_PASSWORD, <password>**
3. Set the properties on the JavaGateway by calling the setProtocolProperties() method, passing the java.util.Properties object.
4. Define the server name as `ssl://<server_name>:<port>`. Set the server name on the ECIRequest object and not on the JavaGateway object.

Configuring SSL for a resource adapter

To configure SSL connection for a resource adapter:

1. Define **ServerName** as `ssl://<server_name>:<port>`.
2. Set the KeyRingClass property to the location of the key ring file.
3. Set the KeyRingPassword property to the password of the key ring file.

Configuring identity propagation

Identity propagation configuration tasks are required on RACF®, CICS Transaction Server and WebSphere Application server. Identity propagation must also be activated in CICS Transaction Gateway. WebSphere Application Server must be configured to specify a user registry to enable user ID and password verification for applications. Prerequisites and configuration tasks are also required for CICS and RACF.

Configuring identity propagation on CICS

The steps required to configure identity propagation on CICS Transaction Server.

CICS Transaction Server requires the following:

- The z/OS identity propagation function provided in z/OS, Version 1.11 or later
- CICS Transaction Server for z/OS Version 4.1 or later with the APAR fixes described in “Configurations that support identity propagation” on page 170. To download these fixes, go to Fix list for CICS Transaction Server for z/OS V4.1
- An IPIC connection with USERAUTH set to IDENTIFY

If the CICS Transaction Gateway making the request and the CICS server are not in the same sysplex (for example when a resource adapter using a local Gateway issues requests directly to CICS), an SSL connection is required to allow the use of USERAUTH=IDENTIFY. For more information, see the “User security” section of “IPIC connection security” on page 159.

Configuring identity propagation on WebSphere Application Server

Configuration is required on WebSphere Application Server to enable identity propagation.

Setting up the identity propagation login module

WebSphere Application Server must be configured to specify a user registry to enable user ID and password verification for applications. Any registry supported by WebSphere Application Server is supported by CICS Transaction Gateway. Examples of the registries supported by WebSphere Application Server are:

- IBM Tivoli Directory Server (ITDS)
- Microsoft Active Directory
- SunOS Directory
- Novel Directory Service

For more information about supported registries, see the WebSphere Application Server Information Center.

All JEE applications that call the CICS Transaction Gateway ECI resource adapter must be configured for container-managed security.

CICS Transaction Gateway includes a JAAS (Java Authentication and Authorization Service) login module in the ECI resource adapter RAR (cicseci.rar). You must install the login module into WebSphere Application Server to enable identity propagation. Install the login module by creating a new JAAS Application Login alias that refers to the fully-qualified name of the login module:
`com.ibm.ctg.security.idprop.LoginModule`

One of the following must be configured to use the CICS Transaction Gateway identity propagation login module:

- The JEE application must be configured to use a custom login configuration that refers to the CICS Transaction Gateway identity propagation login module. This is accessed via the connection factory resource references on the application's configuration panel.
- The connection factory that is used by the application must have a mapping configuration alias that refers to the CICS Transaction Gateway identity propagation login module. This is accessed by the connection factory's configuration panel.

For more information about configuring WebSphere Application Server, see the WebSphere Application Server information center.

Specifying the authentication information to propagate

If identity propagation has been configured and activated, the identity information that can be propagated with a request can be either the identity of the user who invoked the application, or the identity under which the application programmer has configured the application to run.

- The identity of the user who invoked the application is known as the “caller” or “received” identity.
- The identity under which the application programmer has configured the application to run is known as the “run as” or “invocation” identity.

To specify the identity to propagate CICS, you set the propIdentity custom property on the CICS Transaction Gateway identity propagation login module. You do this from the WebSphere Application Server admin console by setting one of the following name-value pairs:

```
propIdentity=Caller
```

or

```
propIdentity=RunAs
```

For example, if you want the “run as” identity to be propagated to CICS, do this:

1. From the WebSphere administrative console; navigate to Security > Global security and select JAAS - Application logins > CTG_idprop > com.ibm.ctg.security.idprop.LoginModule.
2. Ensure the “Use login module proxy” check box is deselected.
3. Select REQUIRED from the “Authentication strategy” drop down list.
4. Create the name-value pair `propIdentity-RunAs`.
5. Click OK.

If you do not specify a setting or if you specify an invalid key or value, the system propagates the “run as” identity by default for application users. The propIdentity key, and the values RunAs and Caller are not case-sensitive.

Configuring identity propagation on RACF

The steps required to configure RACF for identity propagation.

RACF must contain mappings of distinguished names to RACF user IDs. The distinguished names defined in the mappings must have the same format as they have in the user registry.

For more information about configuring IPIC connections and RACF, see the CICS Transaction Server information center.

A command RACMAP is available for creating, deleting, and listing a distributed identity filter. If changes are required, you can delete the filter, and define a new one. The RACMAP command has the following functions:

MAP creates a distributed identity filter

DELMAP
deletes a distributed identity filter

LISTMAP
lists information about a distributed identity filter

Examples:

```
RACMAP ID(GUSKI) MAP
   USERDIDFILTER(NAME('UID=RICH,OU=Web Sales,O=Rich Radio Ham,L=Internet'))
   REGISTRY(NAME('us.richradioham.com'))
   WITHLABEL('Rich''s name filter')
RACMAP ID(SMITH) MAP
  USERDIDFILTER(NAME('uid=JIM,ou=Web Sales,dc=CTGSales, o=HEADOFFICECTG')) -
  REGISTRY(NAME('uk.websales.com'))
```

For more information about the RACMAP command, see the*z/OS Security Server RACF Command Language Reference*.

Note: It is not possible to modify a distributed identity filter.

Configuring identity propagation for CICS Transaction Gateway

Identity propagation must be activated so that CICS Transaction Gateway can flow distributed identities to CICS Transaction Server. Activation involves completing several installation and configuration tasks.

To activate identity propagation for CICS Transaction Gateway:

1. Install a CICS Transaction Gateway ECI resource adapter in WebSphere Application Server. For more information, see “Deploying the CICS resource adapters” on page 36.
2. Configure an IPIC server definition from CICS Transaction Gateway into a CICS server. Alternatively you can configure an IPIC connection from a local Gateway directly into CICS, using SSL.
3. Install the CICS Transaction Gateway identity propagation login module in WebSphere Application Server. For more information, see “Configuring identity propagation on WebSphere Application Server” on page 99.
4. Configure the Java client application resource references, or the connection factories used by the applications, to use the CICS Transaction Gateway identity propagation login module. When applications have been enabled to use the module, identity propagation is active. For more information about configuring WebSphere Application Server, see the documentation for WebSphere Application Server.

Configuring high availability

High availability is supported by the default server, and the CICS request exit.

Configuring a CICS request exit

The **cicsrequestexit** parameter specifies a class that performs dynamic CICS server selection for ECI requests and ESI requests.

You must specify a fully qualified class that implements the com.ibm.ctg.ha.CICSRequestExit interface and must be on the class path of the Gateway daemon. The Gateway daemon supports a single exit at any one time.

For more information see “CICS request exit” on page 189.

Configuring monitoring and statistics

Configuring the request monitoring exits for a Gateway daemon

The **requestexits** parameter specifies a list of one or more classes that perform request monitoring.

Use the configuration tool to set the request monitors to use, or set the **requestexits** parameter to a valid class name for a request monitor class:

```
requestexits=fully_qualified_class_name
```

For example:

```
requestexits=com.ibm.ctg.samples.requestexit.MyMonitor
```

You can define multiple exits by separating them with a comma:

For example:

```
requestexits=com.ibm.ctg.samples.requestexit.1stMonitor,com.ibm.ctg.samples.requestexit.2ndMonitor
```

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Configuring the request monitoring exits for Gateway classes

The configuration for the Gateway classes uses a JVM property, or if using a resource adapter, a custom property.

The JVM property requestExits supports monitoring of base classes without modifying user application code. The resource adapter custom property RequestExits can be configured to allow individual definition of exits for connection factories. The precedents for the two properties are defined in the following table:

Table 4. JVM and resource adapter custom property precedents

JVM property set	Custom property set	Exits loaded from:
No	No	None
Yes	No	JVM property
No	Yes	Custom property
Yes	Yes	Custom property

Set the JVM property to enable a request monitoring exit:

```
-DrequestExits=fully_qualified_class_name
```

For example:

```
java -DrequestExits=com.ibm.ctg.samples.requestexit.BasicMonitor com.ibm.ctg.samples.eci.EciB1
```

You can define multiple exits by separating them with a comma:

```
-DrequestExits=first_exit_name,second_exit_name
```

For example:

```
java -DrequestExits=com.ibm.ctg.samples.requestexit.BasicMonitor,com.ibm.ctg.samples.requestexit.ThreadedMonitor
com.ibm.ctg.samples.eci.EciB1
```

Related information:

"ECI resource adapter deployment parameters" on page 37
The available deployment parameters for the ECI resource adapters and their effect on the final deployed resource adapter. The tools used to configure these parameters are server-specific. The default value is shown where appropriate. Parameters are optional unless indicated as required.

Configuring statistics settings

Use the CICS Transaction Gateway configuration tool to configure the Gateway daemon monitoring resources, or edit the GATEWAY section of the configuration file directly.

Statistics API protocol settings

Use the CICS Transaction Gateway configuration tool to configure the statistics API protocol settings, or edit the statistics API protocol parameters in the GATEWAY section of the configuration file directly.

Bind address:

The **bind** parameter sets the IP address or name of the host to which the protocol handler is to be bound.

If you set a host name in the **Bind address** field of the Configuration tool, it is resolved on startup. If the bind parameter is not specified or is blank, the default behavior is to bind to all IP addresses.

The IP address can be in IPv6 format; for example, `3ffe:307:8:0:260:97ff:fe40:efab`.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Port:

The **port** parameter specifies the number of the TCP/IP port on which the protocol handler listens for incoming client requests.

Set the value in the range 1 through 65,535 in the **Port** field of the Configuration tool.

The default port for TCP/IP is 2006, for SSL the default is 8050, and for the statistics API protocol the default is 2980.

You can override the port setting as follows:

- For TCP/IP use the **ctgstart -port=***number* command
- For SSL use the **ctgstart -sslport=***number* command
- For the statistics API use the **ctgstart -statsport=***number* command

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Connection timeout (ms):

The **connecttimeout** parameter specifies how long the protocol handler waits for a connection manager thread to become available.

Set the value in the range 0 through 65,536 in the **Connection timeout (ms)** field of the Configuration tool, to specify the value in milliseconds. The default is 2000.

When a new connection has been accepted, the protocol handler waits for a connection manager thread to become available. If a connection manager thread does not become available within this time, the connection is refused. If this value is set to zero, a connection is refused if a connection manager thread is not immediately available.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Maximum number of connections:

The **maxconn** parameter specifies the maximum number of applications that can be connected at the same time to perform statistic queries.

Set the value in the **Maximum number of connections** field of the Configuration tool, to the maximum number of connections. The default is 5.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Statistics Interval (HHMMSS)

The **statint** parameter specifies the recording interval for system statistics. The default is three hours. The interval must be at least one minute and cannot be more than 24 hours.

Set the value between 1 minute and 24 hours, including seconds granularity. The **Statistics Interval (HHMMSS)** field of the Configuration tool, field requires the interval to be specified in the format HHMMSS, and within the range 000100 to 240000. The hours (HH) part of the value must be specified in the range 0 to 24, and the minutes (MM) and seconds (SS) part of the value must be specified in the range 00 to 59. The default value is 3 hours (030000). If the value set is less than the minimum it is changed to the minimum, or if the value set is greater than the maximum, it is changed to the maximum. If the value set does not have the correct format, it is changed to the default value.

Set the **statint** parameter in one of these ways:

- Use the Configuration Tool.
- Use the Gateway daemon settings, see Gateway daemon settings.
- Edit the Gateway settings.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Statistics End of Day time (HHMMSS)

The **stateod** parameter specifies the end-of-day time. The End of Day time is used as a point of reference for the clock. Intervals are aligned to this rather than to the CICS Transaction Gateway startup time. This also determines the point at which statistics are reset and potentially recorded, and occurs at least once every 24 hours.

Set the value between 000000 and 235959, in the **Statistics End of Day time (HHMMSS)** field of the Configuration tool, to define the CICS Transaction Gateway End of Day time, in local time. The field requires the interval to be specified in the format HHMMSS and within the range between midnight (000000) and 1 second before midnight (235959). The hours (HH) part of the value must be specified in the range 0 to 23, and the minutes (MM) and seconds (SS) part of the value must be specified in the range 00 to 59. The default is midnight (000000). If the value set is less than the minimum it is changed to the minimum, or if the value set is greater than the maximum, it is changed to the maximum. If the value set does not have the correct format, it is changed to the default value.

Set the **stateod** parameter in one of these ways:

- Use the Configuration Tool.
- Use the Gateway daemon settings; see Gateway daemon settings.

- Edit the Gateway settings in the configuration file; see .

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Configuring the terminal emulator

Perform these tasks to configure the terminal emulator for CICS Transaction Gateway.

Keyboard mapping for cicsterm

The keyboard mapping for terminal emulator operation is defined in a keyboard mapping file.

A sample key mapping file named cicskeysamp.ini is supplied in the <install_path>/samples/configuration subdirectory. It is recommended that you create your own customized mapping file.

The keyboard mapping file can be specified by:

- Using the -k option on the cicsterm command.
- Setting the CICSKEY environment variable. For example:

```
export CICSKEY=/var/cicscli/mykeys.ini
```

If you do not specify otherwise, a file name of cicskey.ini in the <install_path>/bin subdirectory is assumed.

You can change the keyboard mapping file at any time, although changes do not take effect until the next time the terminal emulator is started.

Keyboard mapping file syntax

This section describes the syntax of the keyboard mapping file.

A statement must be provided for each key that is needed, because there are no default assignments (except for the alphabetic and numeric keys). There is no case sensitivity. Each binding must be on a separate line, and of the following form:

```
BIND 3270function [modifier+] key [;comment|#comment]
```

For example, to map the 3270 function EraseEof to the Ctrl+Delete keys pressed together the binding is as follows:

```
bind     EraseEof             Ctrl+Delete ;erase to end of field
```

The keyboard mapping file

In the mapping file, *3270function* can be any one of the following values.

```
backspace       pa1             pf1     pf13
backtab         pa2             pf2     pf14
clear           pa3             pf3     pf15
cursordown                      pf4     pf16
cursorleft      printscreen     pf5     pf17
cursorright     reset           pf6     pf18
cursorselect    tab             pf7     pf19
cursorup                        pf8     pf20
delete          ignore          pf9     pf21
enter                           pf10    pf22
eraseeof                        pf11    pf23
eraseinput                      pf12    pf24
home
insert
newline
```

The value of `ignore` is provided to permit unwanted control keys on the keyboard to be ignored. (Unexpected glyphs are not generated.)

The *Modifier* can be either:

```
Ctrl
Shift
```

The *Key* can be any one of the keys shown in Table 5, (but some combinations of modifier+key are not supported — see "Key combinations"):

Table 5. Keys that you can map

Group	Keys
Escape key	`Escape`
Function keys	`f1 f2 f3 f4 f5 f6 f7 f8 f9 f10 f11 f12`
Numeric keys	`0 1 2 3 4 5 6 7 8 9`
Alphabetic keys	`a b c d e f g h i j k l m` `n o p q r s t u v w x y z`
Tab key	`Tab`

Movement keys	`newline backspace`
	`insert home pageup`
	`delete end pagedown`
	`up-left-down-right`

Keypad keys	`keypad/ keypad* keypad-`
	`keypad7 keypad8 keypad9`
	`keypad4 keypad5 keypad6 keypad+`
	`keypad1 keypad2 keypad3`
	`keypad0 keypad. keypadenter`

Note: For the Client daemon, the `Ctrl/Act`, `Print Screen`, `Scroll Lock` and `Pause` keys are not available. The 3270 function Clear is assigned to the `Esc` key by default.

Key combinations:

The following combinations of modifier and key can be mapped.

No modifier
All keys available for mapping.

Ctrl modifier
Only function keys, movement keys, alphabetic keys, tab key, and keypad keys can be mapped.

Shift modifier
Only function keys, numeric keys, tab key, and alphabetic keys can be mapped.

Customizing the screen colors for cicsterm

Screen colors and attributes are defined in a color mapping file.

A sample color mapping file named cicscolsamp.ini is supplied in the <install_path>/samples/configuration subdirectory. It is recommended that you create your own customized mapping file.

The color mapping file can be identified by:

- Using the -c option on the cicsterm command. For more information see "cicsterm command reference" on page 217.
- Setting the CICSCOL environment variable: For example:

```
export CICSCOL=/var/cicscli/mycols.ini
```

If you do not specify otherwise, a file name of cicscol.ini in the <install_path>/bin subdirectory is assumed.

A color mapping file provides alternative representations in hardware environments where it is not possible exactly to replicate 3270 screen attributes, for example, blinking or underscore. The color mapping file therefore defines how 3270 screen attributes are emulated on the client hardware. If the color mapping file specifies a mapping for an attribute, this mapping is used even if the hardware upon which CICS Transaction Gateway is running actually supports the screen attribute.

The color mapping file defines the default colors to use when there is no color information contained within the 3270 data stream, as well as allowing the remapping of colors. By default, fields in data streams that do not contain any extended attributes are displayed in four colors based on the field intensity and protection attributes. These four colors are defined by the following bind settings in the color mapping file:

normal_unprotected

intensified_unprotected

normal_protected

intensified_protected

By default, fields in data streams that contain extended attributes are displayed in only two colors based on the field intensity attribute. These two colors are defined by the **default** and **default_highlight** bind settings in the color mapping file. You can use the **cicsterm** -e option to specify that the four default colors are used even when the 3270 data stream contains extended attributes.

If an application requests a 3270 field to be displayed with, for example, underscore, and no emulation setting has been specified, and the hardware cannot display underscore, the field is displayed without any highlighting at all.

You can change the color mapping file at any time, although changes do not take effect until the next time the terminal emulator is started.

Color mapping syntax

The syntax of the color mapping file is as follows.

Each binding must be on a separate line, in this form:

```
BIND 3270attrib fg_color[/bg_color] [;comment|#comment]
```

Specifying a space between fg_color and /bg_color causes bg_color to be ignored.

The file is not case-sensitive.

The color mapping file

In the color mapping file, *3270attrib* can be any one of the following values.

```
normal_protected    intensified_protected
normal_unprotected  intensified_unprotected

default    blinking_default    underscored_default
blue       blinking_blue       underscored_blue
green      blinking_green      underscored_green
cyan       blinking_cyan       underscored_cyan
red        blinking_red        underscored_red
magenta    blinking_magenta    underscored_magenta
white      blinking_white      underscored_white
yellow     blinking_yellow     underscored_yellow

default_highlight

operator_information_area
```

Each of *fg_color* and *bg_color* (foreground color and background color) can be any one of the following:

```
black       light_gray
blue        light_blue
brown       yellow
cyan        light_cyan
green       light_green
magenta     light_magenta
red         light_red
gray        white
```

If *bg_color* is omitted, a default value of black is taken.

Configuring trace settings

Use the CICS Transaction Gateway configuration tool to configure the product settings, or edit the trace attributes in the GATEWAY or CLIENT section of the configuration file directly.

To configure the trace settings, select the **Trace Settings** option from the **Tools** menu.

You must restart the Client daemon and the Gateway daemon for any changes to the configuration file to take effect.

Gateway trace file

The **tfile** parameter is the path name of the trace file where Gateway trace messages are written if tracing is enabled.

By default trace output is written to stderr. To specify a file destination, set the **Gateway trace file** field of the Configuration tool. If you specify a file name without a path, the file is created in the `<install_path>/bin` directory.

No trace is written if the CICS Transaction Gateway does not have permission to write to the file you specify. The trace file is overwritten, not appended to, each time the CICS Transaction Gateway starts.

You can also specify a trace file using the tfile option on the **ctgstart** command.

Performance: Turning on the Gateway trace has a significant impact on performance. It is not recommended that Gateway trace is enabled in a production environment.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Gateway trace file wrap size (KB)

The **tfilesize** parameter specifies the size in kilobytes to which the Gateway trace file grows. When the file reaches this size, subsequent trace entries continue to be written from the beginning of the file.

Set the value in the range 0 through 1,000,000 in the **Gateway trace file wrap size (KB)** field of the Configuration tool.

- The default value 0 disables wrapping.
- If you set a value in the range 1 through 39, the CICS Transaction Gateway uses a value of 40 instead, to guarantee an adequate minimum trace size.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Data byte offset in trace data

The **dumpoffset** parameter specifies the byte offset in data to start trace output.

The default value of the **dumpoffset** parameter is zero.

This parameter is in the GATEWAY section of the configuration file. For more information about the other parameters see "GATEWAY section of the configuration file" on page 115

Maximum size of trace data blocks

The **truncationsize** parameter specifies the maximum size of trace data blocks in bytes.

The default value of the **truncationsize** parameter is 80.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Exception stack tracing

The **stack** parameter enables exception stack tracing.

Set the **stack** parameter to **on** to enable exception stack tracing.

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Client trace file

The **tracefile** parameter specifies the path name for the trace file where client trace messages will be written, if tracing is enabled.

You do not have to enter an extension for the file name in the **Client trace file** field of the Configuration tool, because a file of type .BIN is always generated (or .WRP if the trace file wraps).

If no path is specified, the trace is written as follows:

```
/var/cicscli/cicscli.bin
```

To minimize any performance impact, the trace file is written out in binary format. To read it, convert the file to ASCII using the `cicsftrc` command.

For more information about tracing, see "Tracing" on page 245.

This parameter is in the CLIENT section of the configuration file, see "CLIENT section of the configuration file" on page 118 for more information about other parameters in this section.

Client trace file wrap size (KB)

The **maxwrapsize** parameter determines how much storage (KB) is reserved for trace data.

The value is a number in the range 0 - 2,000,000. The default is 0. If you do not specify a value greater than 0, wrapping is disabled. For any value you specify between 1 and 99, 100 KB of storage is reserved by default.

- If you intend using memory mapped tracing you must specify a value greater than 0, otherwise memory mapped tracing is disabled and disk tracing is used instead.
- If you specify a value that is too low and the reserved storage overflows during tracing, subsequent trace records are written to the top of the trace file and overwrite the existing records.

You specify a value for **maxwrapsize** by editing the "CLIENT section of the configuration file" on page 118.

If you are using the Gateway daemon configuration tool, you specify a value for **maxwrapsize** in the **Client trace file wrap size (KB)** field.

Client trace components

The **trace** parameter specifies the Client components to trace when tracing is enabled.

The possible entries for a comma-separated list of components to trace are as follows:

Configuration tool fields	ctg.ini file trace parameters	Description
	ALL	Trace everything
Client API level 1	API	Client API layer level 1
Client API level 2	API.2	Client API layer level 1 and 2
CICSCLI command line	CLI	cicscli command interface
CICSTERM and CICSPRINT	EMU	cicsterm and cicsprnt emulators
CPP classes	CPP	C++ class libraries
Client daemon	CCL	Client daemon
Transport layer level 1	TRN.1	Interprocess communication. The TRN.1 component traces the internal interprocess transport between Client processes. Use it if entries in the Client log refer to functions such as FaarqStart, FaarqStop.
Transport layer level 2	TRN.2	Interprocess communication. The TRN.2 component traces the internal interprocess transport between Client processes. Use it if entries in the Client log refer to functions such as FaarqGetMsg, FaarqPutMsg. TRN.2 is the most verbose tracing component.
Protocol drivers level 1	DRV	Protocol drivers, for example, TCP. This traces data sent and received and provides supplementary information about failures.
Protocol drivers level 2	DRV.2	This traces internal flows through the protocol drivers and interactions with other software components. This enhanced tracking level currently has the same functionality as level 1.

You can also use the `-m` parameter of the `cicscli` command to specify trace components (excluding the Gateway daemon). This overrides any settings in the configuration file. For more information about the cicscli command, see "Administering the Client daemon" on page 204.

You can also use the check boxes in the configuration tool to set trace components.

If you enable tracing without specifying the components in either the cicscli command or in ctg.ini, a default set of components is traced:

- Protocol drivers
- The Client daemon
- Client API level 1
- Transport layer level 1

This parameter is in the CLIENT section of the configuration file, see "CLIENT section of the configuration file" on page 118 for more information about other parameters in this section.

Starting JNI trace

Use one of the methods described here to enable JNI trace.

Use one of the following methods to enable JNI trace:

- Set the following environment variables before you start CICS Transaction Gateway:

  CTG_JNI_TRACE
  Use this environment variable to set the name of the JNI trace file. This environment variable only defines the name of the JNI trace file; it does not enable trace. JNI trace is output as plain text, and there is no requirement to use a particular extension for the file name.

  CTG_JNI_TRACE_ON
  Set this environment variable to YES (case-insensitive) to enable JNI trace when CICS Transaction Gateway is started.
- While CICS Transaction Gateway is running, use the system administration functions. See "Gateway daemon administration" on page 197 for details of how to enable JNI trace dynamically.

If the previous methods are not suitable, use one of the following methods:

- For Java client applications running in local mode, use Java to launch your application and set the system property gateway.T.setJNITFile, as shown in the following example:

  ```
  java -Dgateway.T.setJNITFile=filename application
  ```

  where
  - *filename* is the name of the file to which trace output is to be sent
  - *application* is the application to launch
- When you start CICS Transaction Gateway, issue this command:

  ```
  ctgstart -j-Dgateway.T.setJNITFile=filename
  ```

  where *filename* is the name of the file to which trace output is to be sent. If you do not specify a full path to the file, the location is <install_path>/.

You cannot enable JNI trace through the Configuration Tool.

Configuration parameter reference

The configuration file contains the values used by CICS Transaction Gateway during initialization.

The configuration file

The configuration file contains the values used by CICS Transaction Gateway during initialization.

The configuration file has the default filename `ctg.ini`. .

To create a new configuration file, either use the Configuration Tool, or edit a copy of the sample configuration file supplied in `<install_path>/samples/configuration\ctgsamp.ini` and put it in the `<install_path>/bi` directory.

For more information about the Configuration Tool see "Using the Configuration Tool" on page 44.

The configuration file is a text file that contains a number of sections. Each section begins with SECTION as the first entry on a line, and ends with ENDSECTION. You must define at least one section otherwise CICS Transaction Gateway does not start. The section is typically either a SERVER section or an IPICSERVER section. Any parameters that you do not explicitly define in the configuration file take default values; information about the default values is provided in the individual parameter descriptions.

To denote comments in the configuration file use the hash (#) character. You must put the hash (#) character at the start of the line, or put a space or tab character before it. Some names, for example modename, can begin with a hash (#) character.

There is no restriction on line length in the configuration file.

PRODUCT section of the configuration file

The PRODUCT section of the configuration file defines the applid, applidqualifier, and defaultserver for the Gateway daemon.

Table 6. SECTION PRODUCT

Entry in the configuration file	Description
applid	"Gateway APPLID" on page 45
applidqualifier	"Gateway APPLID qualifier" on page 45
defaultserver	"Default server" on page 48

The following template shows the configuration file definition:

```
SECTION PRODUCT
  APPLID = cccccccc
  APPLIDQUALIFIER = cccccccc
  DEFAULTSERVER=CICSSRV
ENDSECTION
```

If the configuration file (ctg.ini) contains an APPLID but not an APPLID qualifier, the system uses the default value 9UNKNOWN for APPLID qualifier. For more information, see "Gateway APPLID qualifier" on page 45.

Use of the backslash (\) character to split lines in the PRODUCT section of the configuration file is not supported.

GATEWAY section of the configuration file

This table provides the names and descriptions for all parameters that can be set in the GATEWAY section of the configuration file.

There must be one GATEWAY section in the configuration file. Entries correspond to fields in the Gateway daemon settings panel of the Configuration Tool:

Table 7. SECTION GATEWAY

Entry in the configuration file	Description
adminport	"Port for local administration" on page 74.
cicslogging	"Log CICS messages" on page 76
closetimeout	"Timeout for in-progress requests to complete" on page 73.
connectionlogging	"Log Client connections and disconnections" on page 76.
DNSNames	"Display TCP/IP hostnames" on page 76.
dumpoffset	"Data byte offset in trace data" on page 110.
ecigenericreplies	"ECI generic replies" on page 72. Enabling generic replies is not supported.
initconnect	"Initial number of connection manager threads" on page 70.
initworker	"Initial number of worker threads" on page 71.
log@error.dest	"Log destinations" on page 74.
log@error.parameters	"Log file names" on page 74 and "Maximum file sizes" on page 75, and "Maximum number of files" on page 75.
log@info.dest	"Log destinations" on page 74.
log@info.parameters	"Log file names" on page 74, "Maximum file sizes" on page 75 and "Maximum number of files" on page 75.
maxconnect	"Maximum number of connection manager threads" on page 70.
maxworker	"Maximum number of worker threads" on page 71.
msgqualvalidation	"Validate message qualifiers" on page 72. Disabling validation is not supported.
noinput	"Enable reading input from console" on page 71.
notime	"Timing information" on page 77.
quiet	.
requestexits	"Configuring the request monitoring exits for a Gateway daemon" on page 102.
stack	"Exception stack tracing" on page 111.
stateod	"Statistics End of Day time (HHMMSS)" on page 105.
statint	"Statistics Interval (HHMMSS)" on page 105.
tfile	"Gateway trace file" on page 110.
tfilesize	"Gateway trace file wrap size (KB)" on page 110.
truncationsize	"Maximum size of trace data blocks" on page 110.
uowvalidation	"Validate Units of Work" on page 72. Disabling validation is not supported.
workertimeout	"Worker thread availability timeout" on page 73.

If you use the Configuration Tool, do not split long lines. If you do not intend to use the Configuration Tool, you can split long lines by placing the backslash character (\) after a semicolon, as shown:

```
protocol@tcp.parameters=connecttimeout=2000;\
     idletimeout=600000;
```

Two lines are present in the configuration file for each protocol that is enabled. Protocol settings take this form:

- The first line defines the protocol.
- The second line defines the parameters.
- Parameters are separated by a semicolon.

TCP protocol parameters

To enable the TCP protocol, add the name of the TCP protocol handler to the ctg.ini configuration file.

Insert this line:

```
protocol@tcp.handler=com.ibm.ctg.server.TCPHandler
```

Follow it with this:

```
protocol@tcp.parameters=bind=host.domain.org;connecttimeout=3;dropworking;\
idletimeout=4;pingfrequency=5;port=1;requiresecurity;\
solinger=6;
```

Entries for each protocol must be in the form shown:

- Two lines are allowed for each protocol. If you do not intend to use the Configuration Tool, you can split long lines by placing the backslash character \ at the end of a line, to indicate that the following line is a continuation.
- The first line defines the protocol.
- The second line defines the parameters.
- Parameters are separated by a semicolon.

Entries correspond to fields in the TCP settings panel:

Table 8. TCP protocol

Entry in the ctg.ini file	Description
bind	“Bind address” on page 77
connecttimeout	“Connection timeout (ms)” on page 78
dropworking	“Drop working connections” on page 79
idletimeout	“Idle timeout (ms)” on page 78
pingfrequency	“Ping frequency interval (ms)” on page 78
port	“Port” on page 77
requiresecurity	“Require Java Clients to use security classes” on page 79
solinger	“SO_LINGER setting” on page 79

SSL protocol parameters

To enable the SSL protocol, add the name of the SSL protocol handler to the ctg.ini configuration file.

Insert this line:

```
protocol@ssl.handler=com.ibm.ctg.server.SslHandler
```

Follow it with this:

```
protocol@ssl.parameters=clientauth=on;connecttimeout=9;\
dropworking;idletimeout=10;keyring=Key ring or Keystore name;\
keyringpw=a2V5cmluZyBvciBLZXlzdG9yZSBwYXNzd29yZA==;keyringpwscrambled=on;\
pingfrequency=11;port=7;requiresecurity;solinger=12;\
ciphersuites=SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA,SSL_DH_anon_WITH_3DES_EDE_CBC_SHA;
```

Entries for each protocol must be in the form shown:

- Two lines are allowed for each protocol. If you do not intend to use the Configuration Tool, you can split long lines by placing the backslash character \ after a semicolon.
- The first line defines the protocol.
- The second line defines the parameters.
- Parameters are separated by a semicolon.

Entries correspond to fields in the SSL settings panel:

Table 9. SSL protocol

Entry in the configuration file	Description
bind	“Bind address” on page 77
ciphersuites	“Use only these ciphers” on page 83
clientauth	“Use client authentication” on page 82
connecttimeout	“Connection timeout (ms)” on page 78
dropworking	“Drop working connections” on page 79.
idletimeout	“Idle timeout (ms)” on page 78
keyring	“Key ring file” on page 82
keyringpw	“Key ring password” on page 83
keyringpwscrambled	“Key ring password” on page 83.
pingfrequency	“Ping frequency interval (ms)” on page 78
port	“Port” on page 77
requiresecurity	“Require Java Clients to use security classes” on page 79
solinger	“SO_LINGER setting” on page 79

Statistics API protocol parameters

To enable the statistics API protocol, include a protocol handler definition in the GATEWAY section of the configuration file.

The **statsport** parameter in the GATEWAY section of the configuration file is deprecated. If you specify the **statsport** parameter in addition to specifying a statistics API protocol handler definition, the statistics API protocol handler definition takes precedence. You can use the parameter override **-statsport** to override the port number for the statistics API listener port.

To enable the protocol, include a protocol handler definition in the GATEWAY section of the configuration file, for example:

```
protocol@statsapi.handler=com.ibm.ctg.server.RestrictedTCPHandler
protocol@statsapi.parameters=connecttimeout=2000;port=2980;bind=;maxconn=5;
```

Entries for each protocol must be in the form shown:

- Two lines are allowed for each protocol. If you do not intend to use the Configuration Tool, you can split long lines by placing the backslash character \ after a semicolon.

- The first line defines the protocol.
- The second line defines the parameters.
- Parameters are separated by a semicolon.

Table 10. Statistics API protocol parameters

Entry in the ctg.ini file	Description
bind	"Bind address" on page 77
connecttimeout	"Connection timeout (ms)" on page 78
port	"Port" on page 77
maxconn	"Maximum number of connections" on page 74

CLIENT section of the configuration file

The CLIENT section of the configuration file defines the Client daemon settings.

There must be one CLIENT section of the configuration file. Entries correspond to fields in the Client Configuration and Trace settings panels of the Configuration Tool. The section name also stores the value entered in the Application ID field of the Configuration Tool.

Table 11. SECTION CLIENT

Entry in the configuration file	Description
SECTION CLIENT	The Application ID entry is deprecated, it has been replaced by the "Gateway APPLID" on page 45 parameter which now takes precedence. Existing configuration files with the Application ID parameter set are still supported for compatibility with earlier versions. **Note:** SECTION CLIENT must be set to * when creating a new configuration file.
ccsid	CCSID
logfile	"Error and warning log file" on page 88
logfileinfo	"Information log file" on page 88
maxbuffersize	"Maximum buffer size" on page 84
maxrequests	"Maximum requests" on page 85
maxservers	"Maximum servers" on page 85
maxwrapsize	"Client trace file wrap size (KB)" on page 111
printcommand	"Print command" on page 86
printfile	"Print file" on page 86
srvretryinterval	"Server retry interval (Client daemon connections to CICS)" on page 87
terminalexit	"Terminal exit" on page 85
terminstlogging	"Log terminal installations and deletions" on page 88
trace	"Client trace components" on page 112
tracefile	"Client trace file" on page 111

Use of the backslash (\) character to split lines in the CLIENT section of the configuration file is not supported.

IPICSERVER section of the configuration file

The IPICSERVER section of the configuration file defines the CICS server to which the Gateway daemon can connect over IPIC.

The section name also stores the value entered in the Server name field of the Configuration Tool.

Table 12. SECTION IPICSERVER

Entry in the configuration file	Description
SECTION IPICSERVER=<Server name>	"Server name" on page 52
CICSAPPLID	"Target CICS APPLID" on page 54
CICSAPPLIDQUALIFIER	"Target CICS APPLID" on page 54
CONNECTTIMEOUT	"Connection timeout" on page 54
DESCRIPTION	"Description" on page 52
HOSTNAME	"Host name or IP address" on page 53
PORT	"Port" on page 53
SENDSESSIONS	"IPIC send sessions" on page 53
SRVIDLETIMEOUT	"Server idle timeout" on page 55
SRVRETRYINTERVAL	"Server retry interval" on page 54
TCPKEEPALIVE	"Send TCP KeepAlive packets" on page 55

The following template shows the configuration file definition for an IPIC server:

```
SECTION IPICSERVER=PayrollA
  DESCRIPTION=Payroll
  HOSTNAME=cicssrv2.com
  PORT=99
  TCPKEEPALIVE=YES
  SRVIDLETIMEOUT=100
  SRVRETRYINTERVAL=60
  CONNECTTIMEOUT=250
  SENDSESSIONS=100
  CICSAPPLID=CTGTEST1
  CICSAPPLIDQUALIFIER=MYNETWRK
ENDSECTION
```

Use of the backslash (\) character to split lines in the IPICSERVER section of the configuration file is not supported.

SERVER section of the configuration file

The SERVER section of the configuration file defines a server to which the Client daemon can connect.

There might be more than one SERVER section. The first server listed is the default. The section name also stores the value entered in the **Server name** field of the Configuration Tool.

Table 13. SECTION SERVER

Entry in the configuration file	Description
SECTION SERVER	"Server name" on page 57
DESCRIPTION	"Description" on page 52
INITIALTRANSID	"Initial transaction" on page 58

Table 13. SECTION SERVER (continued)

Entry in the configuration file	Description
MODELTERM	"Model terminal definition" on page 58
PROTOCOL	"DRIVER section of the configuration file."
UPPERCASESECURITY	"Use uppercase security" on page 61.

Other entries in this section depend upon the protocol selected.

Table 14. SECTION SERVER: additional entries for the TCP protocol

Entry in the configuration file	Description
CONNECTTIMEOUT	"Connection timeout" on page 59
NETNAME	"Host name or IP address" on page 59
PORT	"Port" on page 59
SRVIDLETIMEOUT	"Server idle timeout" on page 60
TCPKEEPALIVE	"Send TCP/IP KeepAlive packets" on page 60.

Table 15. SECTION SERVER: additional entries for the SNA protocol

Entry in the configuration file	Description
LOCALLUNAME	"Local LU name" on page 68
MODENAME	"Mode name" on page 68
NETNAME	"Partner LU name" on page 68
PARTNERLUALIAS	"Use Partner LU alias name" on page 68.
SRVIDLETIMEOUT	"Server idle timeout" on page 60

Use of the backslash (\) character to split lines in the SERVER section of the configuration file is not supported.

DRIVER section of the configuration file

The DRIVER section of the configuration file determines which communications protocol DLL is used for communicating with a CICS server. The DRIVER section of the configuration file does not have an equivalent configuration panel in the Configuration Tool (the tool automatically selects the correct protocol drivers).

One DRIVER section must exist in the configuration file for each network protocol used with your CICS servers. For more information see "SERVER section of the configuration file" on page 119.

Here is an example of the DRIVER section of the configuration file:

```
SECTION DRIVER=<protocol>
    DRIVERNAME=<library>
ENDSECTION
```

The following table shows valid values for *protocol* and the corresponding DLL:

Table 16. SECTION DRIVER

Valid entries for <*protocol*>	Corresponding DLL
SNA	CCLIBMSN

Table 16. SECTION DRIVER (continued)

Valid entries for <*protocol*>	**Corresponding DLL**
TCPIP	CCLIBMIP

The network protocol must correspond to an entry in the DRIVER section of the configuration file.

Use of the backslash (\) character to split lines in the DRIVER section of the configuration file is not supported.

Summary of environment variables

The table lists the environment variables for controlling how CICS Transaction Gateway functions.

Table 17. Environment variables

Environment variable	**Description**
CICSCLI	Environment variable reference
CTGDCONF	Environment variable reference
CTG_JAVA	Environment variable reference
CTG_JNI_TRACE	"JNI tracing" on page 252
CTG_JNI_TRACE_ON	"JNI tracing" on page 252
LD_PRELOAD	Configuring IBM Communications Server for Linux

Testing your configuration

Run a test to check that CICS Transaction Gateway has been configured correctly.

- Run the JCA resource adapter installation verification test to verify whether the CICS Transaction Gateway ECI resource adapters can be used with your J2EE 1.4 or JEE 5, or later, application server.
- Run the sample programs supplied with CICS Transaction Gateway.

JCA resource adapter installation verification test (IVT)

The JCA resource adapter installation verification test (IVT) verifies whether an ECI resource adapter can be used with a particular application server.

The IVT can be used to verify the use of an ECI resource adapter with an application server as follows:

- The CICS Transaction Gateway V8.1 ECI resource adapter (cicseci.rar) with a JEE 6 certified application server
- Resource adapters supplied in Supportpac CC03 with a J2EE 1.4 certified application server or JEE 5 certified application server

The IVT runs as a servlet within a JEE application server and calls program EC01 on the CICS server. The IVT sends two ECI requests to CICS:

1. A non-transactional request, which is not coordinated by the transaction manager.
2. A transactional request, which uses the global transaction support provided by the application server.

IBM has successfully tested the ECI resource adapters on those application servers listed on the IBM support page. For other JEE application servers, if you experience problems after you have successfully run this IVT, you can report problems to IBM for investigation. If the IVT does not run successfully, problems you encounter are likely to be caused by incorrect deployment of the ECI resource adapter. Investigate the problem using your JEE application server documentation and support organization.

CICS Transaction Gateway Desktop Edition: Support is not provided for the JCA resource adapters.

Prerequisites for running the JCA IVT

Before running the JCA resource adapter installation verification test (IVT) ensure that the JEE application server is compatible, and that the necessary components have been installed and configured correctly. To complete these tasks, you should be able to create deployment plans for the chosen JEE application server, configure CICS Transaction Gateway, and build and install CICS applications.

JEE application server compatibility

The JEE application server you intend using must have passed the J2EE 1.4 or JEE 5 (or later) compatibility test suite. For more information see:

`http://java.sun.com/javaee/overview/compatibility.jsp`

Prerequisites

Complete the following tasks:

- Compile and install the EC01 sample CICS COBOL program on the CICS server.
- Ensure that the CICS Transaction Gateway resource adapter archive RAR file (cicseci.rar) is available.
- Ensure that the JCA IVT enterprise archive (EAR) file is available and has the filename ECIIVT.ear.
- Configure a CICS server connection for CICS Transaction Gateway.

The source for EC01 is shipped with CICS Transaction Gateway as a COBOL file in <install_path>/samples/server/ec01.ccp. Both the RAR files and the EAR files are shipped with the CICS Transaction Gateway in the <install_path>/deployable directory.

To ensure you have correctly configured the CICS Transaction Gateway, follow the instructions in the *CICS Transaction Gateway for Multiplatforms: Application Programming GuideCICS Transaction Gateway for z/OS: Application Programming Guide* to run the EciB1 sample program. The sample program can be found in <install_path>/samples.

Deploying and configuring the JCA IVT

To complete this task you install the resource adapter archive file (.rar), define a connection factory, and set the connection factory properties.

1. Install the ECI resource adapter (cicseci.rar) into your JEE application server. If you want to enable XA support, this can be done by setting the custom property xasupport on the connection factory.
2. Define a connection factory that has the JNDI name set to ECI.
3. Define the connection factory custom properties:

Connection URL
The URL of the CICS Transaction Gateway with which the resource adapter will communicate. In local mode, set the Connection URL to "local:". In remote mode, set the Connection URL to "*protocol*://address", where *protocol* is tcp or ssl.

Port number
In remote mode this is the TCP/IP or SSL port on which the Gateway daemon is configured to listen. Set port number to the port number defined in the configuration file (ctg.ini).

This property is not required in local mode.

Server name
The name of the CICS server to which CICS Transaction Gateway will connect.

- For IPIC in local mode, set the Server Name to "*protocol*:// hostname:port" where *protocol* is tcp or ssl.
- For all other configurations, set the server name to the server defined in the configuration file (ctg.ini).

4. Install the application ECIIVT.ear with a target resource JNDI name of ECIIVTBean1. The ECIIVT.ear is located within the <install_path>/deployable directory.

Running the JCA IVT

To run the JCA IVT.

1. Use a Web browser to display the first IVT Web page index.jsp. For example, on WebSphere point your browser at: http://*app_server_host*:*port*/ECIIVTWeb/ index.jsp
2. Click **Run IVT**.

If the test is successful, it returns a Web page that displays the date and time on the CICS server and a success confirmation message. If the test fails, it returns a Web page with a failure message containing details of the failure including a stack trace option. Capture this data for possible use by the application server support team.

Using the sample programs to check your configuration

After you have configured your system, you can use the sample programs to check that it is configured correctly.

1. Start the CICS Transaction Gateway.
2. Run one of the sample programs supplied. See the *CICS Transaction Gateway for Multiplatforms: Application Programming GuideCICS Transaction Gateway for z/OS: Application Programming Guide* for details, including compilation instructions and information on compiler considerations.

Chapter 6. Scenarios

Follow the steps in these scenarios to learn how to perform tasks such as configuring connections to CICS, or configuring SSL security. As you work through each scenario you use real values provided in a reference table. When you have completed the configuration part of a scenario, you can then test the scenario by sending a simple ECI request to a CICS server.

Sample files

Sample files containing CICS Transaction Gateway configuration parameter values and environment variables are provided for the scenarios. The sample files are installed as part of the product package, and can be accessed from the scenarios through download links.

Each sample file contains values specific to that scenario; The sample files that contain configuration parameters have the common file name ctg.ini. The sample files that contain environment variables have unique filenames such as CTGS05NV. Environment variables are not required for all scenarios.

To open a sample file and view the contents:

1. Open the introductory topic for the scenario, for example "Configuring a secure autoinstalled IPIC connection (SC01)."
2. Scroll down to the sample file link (the link is located immediately below the table of values).
3. Double-click the link.

To download and save a sample file onto your local machine:

1. Right-click the sample file link and select **Save Target As...**.
2. Specify the location where you want to save the sample file.

The samples for UNIX and Windows are installed in the following location:

`<install_path>/samples/scenarios/scnn`

Where `nn` is the scenario number, for example:

`<install_path>/samples/scenarios/sc01`

Configuring a secure autoinstalled IPIC connection (SC01)

You can configure secure autoinstalled IPIC connections using a template. Using a template allows you to change the default connection settings for IPIC autoinstalled connections. To implement an IPCONN template so that IPIC connections are autoinstalled with link security and user security, follow the step-by-step instructions in this scenario.

To configure secure autoinstalled IPIC connections, you must modify the CICS TS sample user-replaceable module (URM) to point to an IPCONN template.

This scenario uses CICS TG connecting to CICS TS V3.2 over IPIC in remote mode. It uses the default name ctg.ini for the configuration file.

Table 18. Values used in this scenario

Component	Parameter	Where set	Example value	Matching values
CICS TG	Server name	IPICSERVER section of ctg.ini	CICSA	
CICS TG	Hostname	IPICSERVER section of ctg.ini	cicssrv2.company.com	
CICS TG	Port 1	IPICSERVER section of ctg.ini	50889	This value must be the same as 3
CICS TS	IPCONN template	In DFHISCIP (autoinstall user program)	SECTEMPL	
CICS TS	TCPIPService 2	TCPIPService definition	Srv50889	This value must be the same as 4
CICS TS	Portnumber 3	TCPIPService definition	50889	This value must be the same as 1
CICS TS	TCPIPService 4	IPCONN definition	Srv50889	This value must be the same as 2
RACF	User ID for link security	IPCONN definition in CICS TS	LINKUSER	
RACF	User ID for user security	Client application	USERID	
RACF	Password for user security	Client application	PASSWORD	

Prerequisites

You must satisfy these system requirements.

Here are the system requirements for CICS TS for z/OS:

- The server must be CICS V3.2 or later because IPIC is not available in earlier releases of CICS.
- TCP/IP services must be active in the CICS server.
  - To activate these services, set the TCP system initialization parameter to YES.
  - To check the status of these services, issue a CEMT INQ TCPIP command and check that the status is open.
- The CICS server must have access to a TCP/IP stack running on the same LPAR.
- The TCP/IP network must extend between LPARs if CICS TG for z/OS and the CICS server exist on different LPARs.
- You must set the SEC system initialization parameter to YES to enable security.
- You must have valid RACF user IDs and passwords.

Here are the system requirements for CICS TG:

- CICS TG must be installed.

To test that the scenario works successfully you can use either the supplied samples, or your own applications. If you use the supplied samples, this scenario requires:

- The sample CICS TG server program EC01 must be compiled, defined, and installed on CICS.
- The CICS TG supplied Java sample EciB2 available on the client machine.

Testing your TCP/IP network

At the transport layer, issue ping requests between the operating system that is hosting your CICS TG and the LPAR where your CICS server resides. The ping request response, as shown in the example below, confirms that the TCP/IP communications are working. The ping request also works if you are using multiple IP stacks on the same LPAR.

```
ping cicssrv2.company.com

Pinging cicssrv2.company.com [1.23.456.789] with 32 bytes of data:

Reply from 1.23.456.789: bytes=32 time<1ms TTL=61
Reply from 1.23.456.789: bytes=32 time<1ms TTL=61
Reply from 1.23.456.789: bytes=32 time<1ms TTL=61
Reply from 1.23.456.789: bytes=32 time<1ms TTL=61

Ping statistics for 1.23.456.789:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Configuring the IPIC server on CICS TG

You must define a server definition for the Gateway daemon to communicate to CICS over IPIC in remote mode.

To define a server definition for the Gateway daemon:

1. Edit the ctg.ini file and define an IPICSERVER definition for your CICS server:
   a. Set HOSTNAME to the name of the z/OS machine that hosts your CICS server.
   b. Set PORT to the port number that your CICS server uses to listen for incoming IPIC requests.

   For example:

   ```
   SECTION IPICSERVER = CICSA
       HOSTNAME=cicssrv2.company.com
       PORT=50889
   ENDSECTION
   ```
2. Save your updated ctg.ini file.
3. Start CICS TG to apply the new IPICSERVER definition.

Configuring the IPCONN autoinstall user program DFHISCIP on CICS TS

To enable the autoinstall of multiple secure IPCONNs, you must modify the sample IPCONN autoinstall program.

CICS provides the IPCONN autoinstall sample program called DFHISxIP in Assembler, C, COBOL, and PL/I , where 'x' denotes the language, which is A, D,

C, and P respectively. The sample program does not use a template by default, so, for autoinstall requests to use a template you must update the program. In this example, the COBOL user program DFHISCIP is updated.

1. Add the MOVE statement to the autoinstall user program DFHISCIP in the A010-INSTALL-IPCONN section. This statement requests CICS to use the IPCONN template SECTEMPL each time the autoinstall user program is called.

```
* - - - - - - - - - -
* Install processing
* - - - - - - - - - -
 A010-INSTALL-IPCONN SECTION.
* Template for secure IPCONN
     MOVE 'SECTEMPL' TO ISAIC-TEMPLATE
```

2. Compile and link-edit your program into a data set that will be picked up by your CICS server.

Configuring the TCPIPSERVICE on CICS TS

The TCPIPSERVICE is a resource that defines the attributes of the IPIC connection, including the listening port and the IPCONN autoinstall user program, referred to as a user replaceable module (URM).

1. Use CEDA to define a TCPIPSERVICE; for example, SRV50889. These values are important:
   - The URM is set to point to your compiled IPCONN autoinstall user program.
   - The port number is set for incoming IPIC requests.
   - The protocol is set to IPIC.
   - The transaction is set to CISS.

   All other values can be left to default. The security section of the TCPIPSERVICE is not applicable for the IPIC protocol; security is applied in the IPCONN definition.

```
CEDA  DEFine TCpipservice( SRV50889 )
 TCpipservice   : SRV50889
 GROup          : HOLLTCPA
 DEscription  ==>
 Urm          ==> DFHISCIP
 POrtnumber   ==> 50889            1-65535
 STatus       ==> Open             Open | Closed
 PROtocol     ==> IPIC             IIop | Http | Eci | User | IPic
 TRansaction  ==> CISS
 Backlog      ==> 00001            0-32767
 TSqprefix    ==>
 Ipaddress    ==>
 SOcketclose  ==> No                No | 0-240000 (HHMMSS)
 Maxdatalen   ==>                   3-524288
```

2. Install the CEDA definition.
3. Check that the TCPIPSERVICE is active. On CICS TS, issue the command:

```
CEMT INQ TCPIPSERVICE
```

   Check these values:
   - The port number shown is correct.
   - The status shows "Ope" for open.
   - The protocol shown is Ipic.
   - The URM shows the IPCONN autoinstall program that you modified.

     For example:

```
CEMT INQ TCPIPSERVICE
STATUS: RESULTS - OVERTYPE TO MODIFY
Tcpips(SRV50889) Ope Por(50889) Ipic Nos Tra(CISS)
Con(00000) Bac( 00128 ) Max( 000000 ) Urm( DFHISCIP )
```

Note: You can configure CICS resources using the CICS Explorer , see the CICS Explorer information in the CICS TS Information Center for more information.

Configuring the IPCONN template on CICS TS

You must define the IPCONN template that each incoming IPIC connection uses. This example implements both link security and user security.

1. Use CEDA to define an IPCONN. The name of the IPCONN must match the name of the template specified in the IPCONN autoinstall user program; for example, SECTEMPL. These values are important:

   TCPIPService
   Set this value to match the name of the TCPIPService defined earlier.

   Receivecount
   Set this value to specify the number of parallel IPCONN sessions.

   SENdcount
   Set this value to zero because IPIC connections are always inbound to CICS TS from CICS TG.

   Inservice
   Set this value to `Yes`.

   Linkauth
   Set this value to `Secuser`.

   SECurityname
   Set this value to an authorized RACF user ID. The user ID must be in a RACF group that is authorized to establish IPIC connections.

   Userauth
   Set this value to `Verify`.

   The APPLID field is relevant only for predefined IPCONN connections. The APPLID field is ignored for autoinstalled IPCONN connections. CICS populates this field with the name of the IPCONN by default.

   This panel is an example of an IPCONN template defined using the CEDA transaction:

```
CEDA  View Ipconn( SECTEMPL )
 Ipconn         : SECTEMPL
 Group          : HOLLIPIC
 Description    :
IPIC CONNECTION IDENTIFIERS
 APplid         : SECTEMPL
 Networkid      :
 Host           :
 (Lower Case)   :
 Port           : No            No | 1-65535
 Tcpipservice   : SRV50889
IPIC CONNECTION PROPERTIES
 Receivecount   : 100           1-999
 SENdcount      : 000           0-999
 Queuelimit     : No            No | 0-9999
 Maxqtime       : No            No | 0-9999
OPERATIONAL PROPERTIES
 AUtoconnect    : No            No | Yes
 Inservice      : Yes           Yes | No
SECURITY
```

```
 SSl            : No             No | Yes
 CErtificate    :                                        (Mixed Case)
 CIphers        :
 Linkauth       : Secuser        Secuser | Certuser
 SECurityname   : LINKUSER
 Userauth       : Verify         Local | Identify | Verify | Defaultuser
RECOVERY
 Xlnaction      : Keep           Keep | Force
```

2. Install the IPCONN definition and check that the output from the CEMT INQ IPCONN(SECTEMPL) command identifies it as INService RELeased.

```
CEMT I IPCONN
 STATUS:  RESULTS - OVERTYPE TO MODIFY
  Ipc(SECTEMPL) App(SECTEMPL) Net(GBIBMIYA) Ins Rel Nos
     Rece(100) Sen(000) Tcp(SRV50889)
```

Testing your scenario

To test that your scenario is configured correctly, use the CICS TG Java sample EciB2 to call CICS server program EC01.

1. To test your scenario using a valid user ID and password, issue the following command from a command prompt on the machine on which CICS TG is running. In this example command, the Gateway daemon TCP handler is listening on the default port.

```
java com.ibm.ctg.samples.eci.EciB2
     jgate=localhost server=CICSA prog0=EC01 commarealength=18
     userid=USERID password=PASSWORD ebcdic
```

   The ebcdic option is not required if you have set up a definition for EC01 in the DFHCNV data conversion macro on CICS.

   The output from the command is as follows:

```
CICS Transaction Gateway Basic ECI Sample 2

 Test Parameters
CICS TG address : localhost:2006
Client security : null
Server security : null
CICS Server : CICSA
UserId : USERID
Password : PASSWORD
Data Conversion : ASCII
Commarea        : null
Commarea length : 18

Number of programs given : 1
  [0] : EC01

Connect to Gateway

Successfully created JavaGateway

CICS servers defined:

System : CICSA

Call Programs

About to call : EC01
  Commarea    :
  Extend_Mode : 0
  Luw_Token   : 0
  Commarea    : 22/05/09 10:05:18
Return code   : ECI_NO_ERROR(0)
Abend code    : null
Successfully closed JavaGateway
```

In the CICS job log you will see this message:

```
DFHIS3000 ... IPCONN 00000006 with applid .00000006 autoinstalled
successfully using autoinstall user program DFHISCIP and template
(SECTEMPL) after a connection request was received on tcpipservice 50889
from host 1.23.456.789
```

where 00000006 is the name of the IPCONN automatically generated by the autoinstall template.

If you issue the command CEMT INQ IPCONN, the output is as follows:

```
CEMT INQ IPCONN
  STATUS:  RESULTS - OVERTYPE TO MODIFY
   Ipc(SECTEMPL) App(SECTEMPL) Net(GBIBMIYA) Ins Rel Nos
      Rece(100) Sen(000) Tcp(SRV50889)
   Ipc(00000001) App(00000001)               Ins Acq Nos
      Rece(100) Sen(000) Tcp(SRV50889)
   Ipc(00000002) App(00000002)               Ins Acq Nos
      Rece(100) Sen(000) Tcp(SRV50889)
```

The example shows two active IPCONN connections autoinstalled from different Gateway daemons. Note that the IPCONN autoinstall template remains INS REL.

2. If you test your scenario using an incorrect user ID and password combination, you receive an ECI_ERR_SECURITY_ERROR RC=27 message. In the CICS job log, the following message is displayed:

   ```
   DFHIS1027 ... Security violation has been detected using IPCONN 0000006 and
   transaction id CPMI by userid CICSUSER
   ```

Optional: using the APPLID to identify your CICS TG

To identify your CICS TG to CICS when connecting over IPIC, you can provide your APPLID in the ctg.ini file or specify an APPLIDQUALIFIER and the APPLID.

To provide your APPLID and APPLIDQUALIFIER in the ctg.ini file, specify:

```
SECTION PRODUCT
    APPLID=MYAPPL
    APPLIDQUALIFIER=MYQUAL
ENDSECTION
```

If you use an APPLID of MYAPPL and an APPLIDQUALIFIER of MYQUAL, the CICS system log shows the following messages when an IPCONN is installed:

```
DFHIS3000 .... IY2GTGA2 IPCONN APPL with applid MYQUAL.MYAPPL
autoinstalled successfully using autoinstall user program DFHISCIP
and template SECTEMPL after a connection request was received on
tcpipservice SRV50889 from host 1.23.456.789

DFHIS2001 .... IY2GTGA2 Client session from applid MYAPPL accepted for IPCONN
APPL.
```

By default, the user replaceable module DFHISCIP uses the last four characters of the incoming CICS TG APPLID as the name of the IPCONN. In this example, the last four characters of MYAPPL are APPL because padded spaces are ignored.

To view the installed IPCONN (APPL) and the template (SECTEMPL), issue the CEMT INQ IPCONN command:

```
CEMT INQ IPCONN
STATUS:  RESULTS - OVERTYPE TO MODIFY
 Ipc(APPL    ) App(MYAPPL  ) Net(MYQUAL  ) Ins Acq Nos
    Rece(100) Sen(000) Tcp(SRV50889)
 Ipc(SECTEMPL) App(SECTEMPL) Net(GBIBMIYA) Ins Rel Nos
    Rece(100) Sen(000) Tcp(SRV50889)
```

Note that the IPCONN template must be INS REL for it to be used by an incoming request. The autoinstalled IPCONN, for example, APPL, is INS ACQ.

Configuring a secure predefined IPIC connection (SC02)

A predefined IPCONN provides a more secure environment and can prevent unwanted IPCONN autoinstall requests from succeeding. To configure a secure predefined IPCONN for your IPIC connection between CICS TG and CICS TS, follow the step-by-step instructions in this scenario.

This scenario uses CICS TG connecting to CICS TS V3.2 over IPIC in remote mode. It uses the default name ctg.ini for the configuration file.

Table 19. Values used in this scenario

Component	Parameter	Where set	Example value	Matching values
CICS TG	APPLID 1	PRODUCT section of ctg.ini	MYAPPL	This value must be the same as 6
CICS TG	APPLIDQUALIFIER 2	PRODUCT section of ctg.ini	MYQUAL	This value must be the same as 7
CICS TG	Server name	IPICSERVER section of ctg.ini	CICSA	
CICS TG	Hostname	IPICSERVER section of ctg.ini	cicssrv2.company.com	
CICS TG	Port 3	IPICSERVER section of ctg.ini	50889	This value must be the same as 5
CICS TS	TCPIPService 4	TCPIPService definition	Srv50889	This value must be the same as 8
CICS TS	Portnumber 5	TCPIPService definition	50889	This value must be the same as 3
CICS TS	APPLID 6	IPCONN definition	MYAPPL	This value must be the same as 1

Table 19. Values used in this scenario (continued)

Component	Parameter	Where set	Example value	Matching values
CICS TS	Network ID **7**	IPCONN definition	MYQUAL	This value must be the same as **2**
CICS TS	TCPIPService **8**	IPCONN definition	Srv50889	This value must be the same as **4**
RACF	User ID for link security	IPCONN definition in CICS TS	LINKUSER	
RACF	User ID for user security	Client application	USERID	
RACF	Password for user security	Client application	PASSWORD	

Prerequisites

You must satisfy these system requirements.

Here are the system requirements for CICS TS for z/OS:

- The server must be CICS V3.2 or later because IPIC is not available in earlier releases of CICS.
- TCP/IP services must be active in the CICS server.
  - To activate these services, set the TCP system initialization parameter to YES.
  - To check the status of these services, issue a CEMT INQ TCPIP command and check that the status is open.
- The CICS server must have access to a TCP/IP stack running on the same LPAR.
- The TCP/IP network must extend between LPARs if CICS TG for z/OS and the CICS server exist on different LPARs.
- You must set the SEC system initialization parameter to YES to enable security.
- You must have valid RACF user IDs and passwords.

Here are the system requirements for CICS TG:

- CICS TG must be installed.

To test that the scenario works successfully you can use either the supplied samples, or your own applications. If you use the supplied samples, this scenario requires the following:

- The sample CICS TG server program EC01 must be compiled, defined, and installed on CICS.
- The CICS TG supplied Java sample EciB2 available on the client machine.

Testing your TCP/IP network

At the transport layer, issue ping requests between the operating system that is hosting your CICS TG and the LPAR where your CICS server resides. The ping request response, as shown in the example, confirms that the TCP/IP

communications are working. The ping request also works if you are using multiple IP stacks on the same LPAR.

```
ping cicssrv2.company.com

Pinging cicssrv2.company.com [1.23.456.789] with 32 bytes of data:

Reply from 1.23.456.789: bytes=32 time<1ms TTL=61
Reply from 1.23.456.789: bytes=32 time<1ms TTL=61
Reply from 1.23.456.789: bytes=32 time<1ms TTL=61
Reply from 1.23.456.789: bytes=32 time<1ms TTL=61

Ping statistics for 1.23.456.789:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Configuring the IPIC server on CICS TG

You must edit the ctg.ini file to identify your CICS TG to CICS and to define a server definition for the Gateway daemon to communicate with CICS over IPIC in remote mode.

1. To identify your CICS TG to CICS when connecting over IPIC, you must define your APPLID and APPLIDQUALIFIER, in uppercase, in the PRODUCT section of the ctg.ini file.

   For example:

   ```
   SECTION PRODUCT
       APPLID=MYAPPL
       APPLIDQUALIFIER=MYQUAL
   ENDSECTION
   ```

2. To define an IPICSERVER definition for your CICS server:
   a. Set HOSTNAME to the TCP/IP host name or TCP/IP address on which CICS is listening.
   b. Set PORT to the port number that your CICS server uses to listen for incoming IPIC requests.

   For example:

   ```
   SECTION IPICSERVER = CICSA
       HOSTNAME=cicssrv2.company.com
       PORT=50889
   ENDSECTION
   ```

3. Save your updated ctg.ini file.
4. Start CICS TG to apply the new definitions.

Configuring the TCPIPService on CICS TS

The TCPIPService is a resource that defines the attributes of the IPIC connection, including the listening port.

1. Use CEDA to define a TCPIPService; for example, SRV50889. These values are important:
   - The URM is set to NO to prevent the default IPCONN autoinstall program from running.
   - The port number is set for incoming IPIC requests.
   - The protocol is set to IPIC.
   - The transaction is set to CISS.

All other values can be left to default. The security section of the TCPIPService is not applicable for the IPIC protocol; security is applied in the IPCONN definition.

```
CEDA  DEFine TCpipservice( SRV50889  )
       TCpipservice  : SRV50889
       GROup          : HOLLIPIC
       DEscription  ==>
       Urm          ==> NO
       POrtnumber   ==> 50889     1-65535
       STatus       ==> Open      Open | Closed
       PROtocol     ==> IPIC      IIop | Http | Eci | User | IPic
       TRansaction  ==> CISS
       Backlog      ==> 00001     0-32767
       TSqprefix    ==>
       Ipaddress    ==>
       SOcketclose  ==> No                No | 0-240000 (HHMMSS)
       Maxdatalen   ==>                   3-524288
```

2. Install the CEDA definition.
3. Check that the TCPIPService is active. On CICS TS, issue the command:

   ```
   CEMT INQ TCPIPSERVICE
   ```

   Check the following values:
   - The port number shown is correct.
   - The status shows "Ope" for open.
   - The protocol shown is Ipic.
   - The URM shows NO to state that IPCONN autoinstall is not permitted on this TCPIPSERVICE.

     For example:

     ```
     CEMT INQ TCPIPSERVICE
     STATUS: RESULTS - OVERTYPE TO MODIFY
     Tcpips(SRV50889) Ope Por(50889) Ipic Nos Tra(CISS)
     Con(00000) Bac( 00128 ) Max( 000000 ) Urm(NO        )
     ```

Configuring the IPCONN on CICS TS

You must define the IPCONN for the incoming IPIC connection. This example implements both link security and user security.

1. Use CEDA to define an IPCONN. These values are important:

   APplid
   Set this value to match the APPLID specified in the ctg.ini file.

   Networkid
   Set this value to match the APPLIDQUALIFIER specified in the ctg.ini file.

   TCPIPService
   Set this value to match the name of the TCPIPService defined earlier.

   Receivecount
   Set this value to specify the number of parallel IPCONN sessions.

   SENdcount
   Set this value to zero because IPIC connections are always inbound to CICS TS from CICS TG.

   Inservice
   Set this value to `Yes`.

Linkauth
Set this value to `Secuser`.

SECurityname
Set this value to an authorized RACF user ID. The user ID must be in a RACF group that is authorized to establish IPIC connections.

Userauth
Set this value to `Verify`.

Leave all the other values to default.

This panel is an example of an IPCONN definition defined using the CEDA transaction:

```
CEDA  View Ipconn( IPC50889 )
       Ipconn         : IPC50889
       Group          : HOLLIPIC
       Description    :
      IPIC CONNECTION IDENTIFIERS
       APplid         : MYAPPL
       Networkid      : MYQUAL
       Host           :
       (Lower Case)   :
       Port           : No       No | 1-65535
       Tcpipservice   : SRV50889
      IPIC CONNECTION PROPERTIES
       Receivecount   : 100      1-999
       SENdcount      : 000      0-999
       Queuelimit     : No       No | 0-9999
       Maxqtime       : No       No | 0-9999
      OPERATIONAL PROPERTIES
       AUtoconnect    : No       No | Yes
       Inservice      : Yes      Yes | No
      SECURITY
      SSl            : No       No | Yes
       CErtificate    :                        (Mixed Case)
       CIphers        :
       Linkauth       : Secuser  Secuser | Certuser
       SECurityname   : LINKUSER
       Userauth       : Verify   Local | Identify | Verify | Defaultuser
      RECOVERY
       Xlnaction      : Keep     Keep | Force
```

2. Install the IPCONN definition and check that the output from the CEMT INQ IPCONN(IPC50889) command identifies it as INService RELeased.

```
CEMT I IPCONN
       STATUS:  RESULTS - OVERTYPE TO MODIFY
        Ipc(IPC50889) App(MYAPPL  ) Net(MYQUAL  ) Ins Rel Nos
           Rece(100) Sen(000) Tcp(SRV50889)
```

Testing your scenario

To test that your scenario is configured correctly, use the CICS TG Java sample EciB2 to call CICS server program EC01.

1. To test your scenario using a valid user ID and password, issue the following command from a command prompt on the machine on which the CICS TG is running. In this example command, the Gateway daemon TCP handler is listening on the default port.

```
java com.ibm.ctg.samples.eci.EciB2
     jgate=localhost server=CICSA prog0=EC01 commarealength=18
     userid=USERID password=PASSWORD ebcdic
```

The ebcdic option is not required if you have set up a definition for EC01 in the DFHCNV data conversion macro on CICS.

The output from the command is as follows:

```
      CICS Transaction Gateway Basic ECI Sample 2

       Test Parameters
      CICS TG address : localhost:2006
Client security : null
Server security : null
CICS Server : CICSA
UserId : USERID
Password : PASSWORD
Data Conversion : ASCII
      Commarea        : null
      Commarea length : 18

      Number of programs given : 1
        [0] : EC01
      Connect to Gateway
      Successfully created JavaGateway

      CICS servers defined:
      System : CICSA

      Call Programs
      About to call : EC01
        Commarea    :
        Extend_Mode : 0
        Luw_Token   : 0
        Commarea    : 24/06/09 11:17:19
      Return code   : ECI_NO_ERROR(0)
      Abend code    : null
      Successfully closed JavaGateway
```

In the CICS job log you will see this message:

```
DFHIS2001 ... Client session from applid MYAPPL accepted for
IPCONN IPC50889.
```

Issuing CEMT INQ TCPIPSERVICE shows that the connection count has increased to 1.

```
CEMT INQ TCPIPSERVICE
         STATUS: RESULTS - OVERTYPE TO MODIFY
          Tcpips(SRV50889) Ope Por(50889) Ipic Nos Tra(CISS)
           Con(00001) Bac( 00128 ) Max( 000000 ) Urm(NO        )
```

The IPCONN connection remains established until the connection is explicitly released, either by CICS TS or CICS TG.

2. If you test your scenario using an incorrect user ID and password combination, you receive an ECI_ERR_SECURITY_ERROR RC=27 message. In the CICS job log, the following message is displayed:

   ```
   DFHIS1027 ... Security violation has been detected using
   IPCONN IPC50889 and transaction id CPMI by userid CICSUSER
   ```

3. If your APPLID and APPLIDQUALIFIER specified in the ctg.ini file do not match the APPLID and NETWORKID defined on the IPCONN, your IPCONN connection will not be established; CICS TS will then attempt to autoinstall your IPCONN connection. However, because autoinstall is not enabled (the TCPIPService has URM specified as NO) the autoinstall is rejected and your ECI request causes a program abend with an ECI_ERR_NO_CICS(-3) message. In the CICS job log, you see this message:

   ```
   DFHIS3001 ... IPCONN autoinstall rejected after a connection
   was received on TCPIPSERVICE SRV50889 from host 1.23.456.789
   because the TCPIPSERVICE has URM(NO)
   ```

Optional: specifying CICSAPPLID and CICSAPPLIDQUALIFIER in the IPICSERVER definition

To ensure that your CICS TG connects to the expected CICS server, you can specify CICSAPPLID and CICSAPPLIDQUALIFIER in the IPICSERVER definition in the ctg.ini file.

1. Add your CICSAPPLID and CICSAPPLIDQUALIFIER definitions in the IPICSERVER section.

   For example:

```
SECTION IPICSERVER = A1-IPIC
          SRVIDLETIMEOUT=0
          HOSTNAME=cicssrv2.company.com
          PORT=50889
          CONNECTTIMEOUT=60
          TCPKEEPALIVE=Y
          SENDSESSIONS=100
          CICSAPPLID=IY2GTGA2
          CICSAPPLIDQUALIFIER=GBIBMIYA
       ENDSECTION
```

2. Save your updated ctg.ini file.
3. Start CICS TG to apply the new definitions.

If the CICSAPPLID and CICSAPPLIDQUALIFIER in your ctg.ini file do not match the APPLID and network ID of your CICS server as defined in the CICS System Initialization Table (SIT), your ECI request causes a program abend with an `ECI_ERR_NO_CICS(-3)` message. In the CICS job log, you see this message:

```
DFHIS1013 ... Invalid applid GBIBMIYA.IY2GTGXX received in capability exchange
request on TCPIPSERVICE SRV50889.
```

Configuring SSL between a Java client and CICS TG (SC06)

This scenario shows how to configure SSL on the connection between a Java client running in Windows and CICS Transaction Gateway for Multiplatforms. The connection between CICS Transaction Gateway and CICS Transaction Server for z/OS is over TCP/IP.

In this scenario you configure SSL security on the Gateway daemon, configure SSL server authentication and (optionally) SSL client authentication, enable SSL, and send an ECI request to the CICS server to check that the SSL connection works.

In this scenario server authentication occurs on CICS Transaction Gateway, and optional client authentication occurs on the Java client.

The following figure shows the topology used in this scenario.

Follow the step-by-step instructions in this scenario using the following values:

Component	Parameter	Where set	Example value
CICS TG	protocol@ssl.handler	SECTION GATEWAY in ctg.ini	com.ibm.ctg.server.SslHandler
CICS TG	clientauth	SECTION GATEWAY in ctg.ini	on
CICS TG	keyring	SECTION GATEWAY in ctg.ini	MyServer.jks
CICS TG	keyringpw	SECTION GATEWAY in ctg.ini	MyPassword
CICS TG	port	SECTION GATEWAY in ctg.ini	8573
CICS TG	SERVER	SECTION SERVER in ctg.ini	CICSA
CICS TG	PROTOCOL	SECTION SERVER in ctg.ini	TCPIP
CICS TG	NETNAME	SECTION SERVER in ctg.ini	cicssrv1.company.com
CICS TG	PORT	SECTION SERVER in ctg.ini	7760

The following sample configuration file for this scenario is available for you to download:

- ctg.ini

Prerequisites for the SSL scenario

Before you can complete this scenario, you must ensure that the system requirements for CICS Transaction Server and CICS Transaction Gateway are satisfied.

CICS Transaction Server on z/OS:

- A working connection from CICS Transaction Gateway to CICS is required. This can be an IPIC, SNA, or TCP/IP connection. This scenario uses a TCP/IP connection to CICS. For more information see “Configuring TCP/IP” on page 55.
- TCP/IP services must be active in the CICS server.
  - To activate these services, set the TCP system initialization parameter to YES.
  - To check the status of these services, issue a CEMT INQ TCPIP command and check that the status is open.
- The CICS server must have access to a TCP/IP stack running on the same LPAR.

CICS Transaction Gateway:

- CICS Transaction Gateway must be correctly installed.

To test the scenario works successfully you can either use the supplied samples, or your own applications. If you choose to use the supplied samples, this scenario requires:

- The sample CICS TG server program EC01 to be compiled, defined, and installed on CICS.
- The CICS Transaction Gateway supplied Java sample EciB1 be available on the Java client machine.

Configuring SSL server authentication

To complete this task you use iKeyman to create a server keyring and a server certificate. You then use iKeyman to export the certificate, create a client keyring, and import the server certificate into the keyring.

iKeyman is installed in: <install_path>/jvm160/bin

For information about the benefits of using SSL see "Why use SSL?" on page 161.

UNIX and Linux commands are case-sensitive; on these platforms when starting the iKeyman tool, issue the command like this: `ikeyman`.

Create a server keyring

The keyring contains your server certificate and its associated private key. SSL uses the certificate to identify the server to connecting clients.This keyring must be used exclusively on the server and must be kept secure.

1. Start iKeyman.
2. On the iKeyman main menu, click **Key database file** > **New**.
3. On the **Key database type** menu, select **JKS**.
4. In the **File name** field, type a name for your keyring, for example *MyServer.jks*.
5. In the **Location** field, type the path where you want to store the server keyring.
6. Click **OK**.
7. Type the password for accessing the keyring file. This scenario uses the password `MyPassword`.
8. Click **OK**.

Create a server certificate

Now you are ready to create the self-signed server certificate and store it with its private key in the server keyring:

1. On the iKeyman main menu, click **Create** > **New Self-Signed Certificate**.
2. In the New self-signed certificate window, complete the following steps:
   a. In the **Key label** field type `exampleservercert`.
   b. On the **Version** menu, select **X509 V3**.
   c. On the **Key size** menu, select **1024**.

      The **common name** defaults to the name of your machine, and the **validity period** defaults to 365 days.
3. Click **OK**.

   iKeyman now generates a public/private key pair, and an entry for the `exampleservercert` certificate you have just created appears in the Personal Certificates window.
4. Select the `exampleservercert` certificate and click **View/Edit**.

   The Key information window for the certificate opens. The information in the **Issued to** (certificate requester) and **Issued by** (signer) text boxes is identical.

   To establish an SSL connection with a server that presents this certificate, the client must trust the signer. To do this the client key repository must contain the signer certificate of the server that presents the *exampleservercert* certificate.

Export the server signer certificate

1. Select the *exampleservercert* certificate and click **Extract Certificate**.
2. On the **Data type** menu, select **Base64-encoded ASCII**.
3. In the **Certificate file name** field, type the name of the text file that contains your server certificate data `exampleservercert.arm`.
4. In the **Location** field, type the type the path where you want to store the certificate file.
5. Click **OK**.

The exported certificate is a signer certificate generated from the personal certificate in the keyring, it does not contain the private key. Import the certificate into the keyring of any client that needs to communicate with this SSL server. The certificate allows the client to verify the identity of the server.

Create a client keyring

A client keyring must contain, as a minimum, the signer certificate of the SSL server keyring. This keyring is used by the client application, to verify the identity of the server. If client authentication is required it must also contain a client personal certificate, used to prove its own identity. For more information see Configuring SSL client authentication.

To create a client keyring:

1. Start iKeyman.
2. On the iKeyman main menu click **Key Database File** > **New**.
3. On the **Key Database Type** menu, select **JKS**.
4. In the **File name** field, type the client keyring file name, for example `MyClient.jks`.
5. In the **Location** field, type the path where you want to store the client keyring.
6. Click **OK**.
7. Type a password for accessing the keyring. This scenario uses the password `MyPassword`.
8. Click **OK**.

Import the server signer certificate

1. In the **Signer certificates** list, select the certificate name **exampleservercert.arm**.
2. Click **Add**.
3. In the Certificate file name field type a unique, recognizable name, for example, `my self-signed server authority`.
4. Click **OK**.

   The new signer certificate is added to the **Signer Certificates** list and can be used by the client application to verify the identity of the server.

You have now configured SSL server authentication.

Configuring SSL client authentication (optional)

To complete this task you use iKeyman to create a client certificate and export the client certificate. You then use iKeyman to import the certificate and a public (CA) certificate into the server keyring.

iKeyman is installed in <install_path>/jvm160/bin

SSL client authentication is an option that provides extra security by determining which client applications are allowed to connect to the Gateway daemon. This builds on the security provided by SSL server authentication.

If the SSL handler used by the CICS Transaction Gateway is configured to support server but not client authentication, you do not need to create a client certificate as described here because the client keyring requires just the signer certificate of the server, which you have already imported.

Create a client certificate

For client authentication to occur, the client keyring must contain a self-signed certificate that is used for identifying the connecting client to the server.

1. Start iKeyman.
2. On the certificates menu, click **Personal Certificates**.
3. Click **Create** > **New Self-Signed Certificate**.
4. In the Create New Self-Signed Certificate window, complete the following steps:
   a. In the **Key label** field, type `exampleclientcert`.
   b. On the **Version** menu, select **X509 V3**.
   c. On the **Key size** menu, select **1024**.

      The **Common name** defaults to the name of the machine you are using, and the **Validity period** defaults to 365 days.
5. Click **OK**.

   iKeyman now generates a public/private key pair, and an entry for the `exampleclientcert` certificate you have just created appears in the Personal Certificates window.

Export the client signer certificate

1. In the certificate list, select **exampleclientcert** and click **Extract Certificate**.
2. On the **Data type** menu, select **Base64-encoded ASCII**.
3. In the **Certificate file name** field, type the name of the text file containing the client certificate `exampleclientcert.arm`.
4. Click **OK**.

The exported certificate is a signer certificate generated from the personal certificate in the keyring, it does not contain the private key. Import it into the keyring of all servers that need to communicate with the SSL client. This certificate allows the server to verify the identity of the client.

Import the client signer certificate

1. On the iKeyman main menu click **Key Database File** > **Open**.
2. Select **MyServer.jks**.
3. In the Signer Certificates view, select **Add**.
4. Locate the stored Server Base64-encoded ASCII certificate file `exampleclientcert.arm`.
5. Click **OK**.
6. Give this signer certificate the unique label `My Self-Signed Client Certificate`.
7. Select **OK**.

The new signer certificate is added to the list in the Signer Certificates view, and can now be used by the server to verify the identity of the client application.

Configuring the Gateway daemon for SSL

To complete this task you edit the CICS Transaction Gateway configuration file (ctg.ini) to define the SSL protocol handler and its parameters.

The Gateway daemon requires details of the server keyring `MyServer.jks`. This keyring contains the server certificate `exampleservercert` that the Gateway daemon SSL handler uses as a personal certificate to identify itself to the client.

If client authentication is enabled, the server keyring requires the client certificate as a signer certificate. In this scenario, the client certificate is `exampleclientcert` and in the server keyring, `My Self-Signed Client Certificate`. The Gateway daemon SSL handler uses this signer certificate to verify the identity of the client when it attempts to connect using its personal certificate.

1. Edit the ctg.ini configuration file to add the following SSL protocol handler definition:

   ```
   protocol@ssl.handler=com.ibm.ctg.server.SslHandler
   ```

2. Complete the following tasks:
   a. Set **clientauth** to `on`. Do this if you followed the steps on Configuring SSL client authentication.

      This parameter determines whether or not client authentication occurs. Valid values are `on` (client authentication occurs) and `off` (client authentication does not occur). The default is `off`.

   b. Set **keyring** to `MyServer.jks`.

      This is the name of the keyring to be used by this SSL protocol handler. The keyring must be accessible by the user ID under which the Gateway daemon is running. The value must be either the full path name, or the relative path name of the keyring file. Relative path names are resolved relative to <install_path>/bin. Use either a forward slash (/) character or double backslash (\\) characters as a separator in the path name on all operating systems.

   c. Set **keyringpw** to`MyPassword`.

      This is the password that you used for the server key ring..

   d. Set **port** to 8573.

      This parameter identifies the TCP/IP port on which the protocol handler listens for incoming client requests.

   When you have completed these steps the SSL protocol handler definition should look like this:

   ```
   protocol@ssl.parameters=clientauth=on;keyring=MyServer.jks;
   keyringpw=MyPassword;port=8573
   ```

3. Save the changes.

You have now configured the Gateway daemon for SSL.

Verifying that SSL is enabled on the connection

To complete this task you start CICS Transaction Gateway and check the messages that confirm SSL is enabled.

Start CICS Transaction Gateway from the command line:

```
ctgadmin -a start
```

If the SSL protocol handler starts successfully CICS Transaction Gateway generates two messages:

- The first message lists the SSL ciphers that have been enabled, for example:

```
CTG8401I The following ciphers are enabled:
 SSL_RSA_WITH_RC4_128_MD5
 SSL_RSA_WITH_RC4_128_SHA
 SSL_RSA_WITH_AES_128_CBC_SHA
 SSL_DHE_RSA_WITH_AES_128_CBC_SHA
 SSL_DHE_DSS_WITH_AES_128_CBC_SHA
 SSL_RSA_WITH_3DES_EDE_CBC_SHA
 SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA
 SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
 SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA
 SSL_DHE_DSS_WITH_RC4_128_SHA
 SSL_RSA_WITH_DES_CBC_SHA
 SSL_RSA_FIPS_WITH_DES_CBC_SHA
 SSL_DHE_RSA_WITH_DES_CBC_SHA
 SSL_DHE_DSS_WITH_DES_CBC_SHA
 SSL_RSA_EXPORT_WITH_RC4_40_MD5
 SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
 SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
 SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
```

- The second message confirms that the SSL protocol handler started successfully and identifies the port being used, for example:

```
CTG6524I Successfully started handler for the ssl: protocol on port 8573
```

If the SSL protocol handler fails to start, CICS Transaction Gateway generates a message explaining the reason, for example:

```
CTG6525E Unable to start handler for the ssl: protocol, port: 8573,
because: invalid port number
```

If a Java exception has occurred, rectify the problem, restart CICS Transaction Gateway and check that the protocol handler has started.

You have now verified that SSL is enabled on the connection.

Testing the SSL scenario

To complete this task you set the Java CLASSPATH environment variable, then issue a Java command that invokes the EciB1 sample application to send an ECI request to CICS.

Set the Java CLASSPATH variable

The Java CLASSPATH environment variable identifies the location of the `ctgclient.jar` and `ctgsamples.jar` files.

1. Open a command prompt window and change to the directory where the Java keystore file is located.
2. Set the Java CLASSPATH environment variable with the export CLASSPATH command, for example:

```
export CLASSPATH=<install_path>/classes/ctgclient.jar:
<install_path>/classes/ctgsamples.jar:$CLASSPATH
```

Send an ECI request to CICS

To send an ECI request to CICS you issue a Java command that calls the EciB1 sample application, specifying the ssl:// protocol. When you do this the Java client and the Gateway daemon attempt an SSL handshake. If server authentication is

successful, and if client authentication (if configured) is successful, the Gateway daemon lists the available CICS servers. You then select the CICS server. The CICS application EC01 then confirms the request by returning the current date and time.

The source for the sample EciB1 is located in the samples folder:

```
/opt/ibm/samples/java/com/ibm/ctg/samples/eci
```

1. Start CICS Transaction Gateway from a command line prompt:

   ```
   ctgadmin -a start
   ```

2. Enter the Java command that calls the EciB1 sample application using the following format:

   ```
   java com.ibm.ctg.samples.eci.EciB1 ssl://Gateway_URL
   Gateway_port_number jks_filename jks_password
   ```

   For example:

   ```
   java com.ibm.ctg.samples.eci.EciB1 ssl://cicssrv1.company.com 8573 MyClient.jks
   MyPassword
   ```

   CICS Transaction Gateway returns details of the available CICS servers, for example:

   ```
   CICS Servers Defined:

            1. CICSA -CICS V4.1 Server

   Choose Server to connect to, or q to quit:
   ```

3. Enter the number of the CICS server where you want to send the ECI request.

The specified CICS server returns the current date and time, for example:

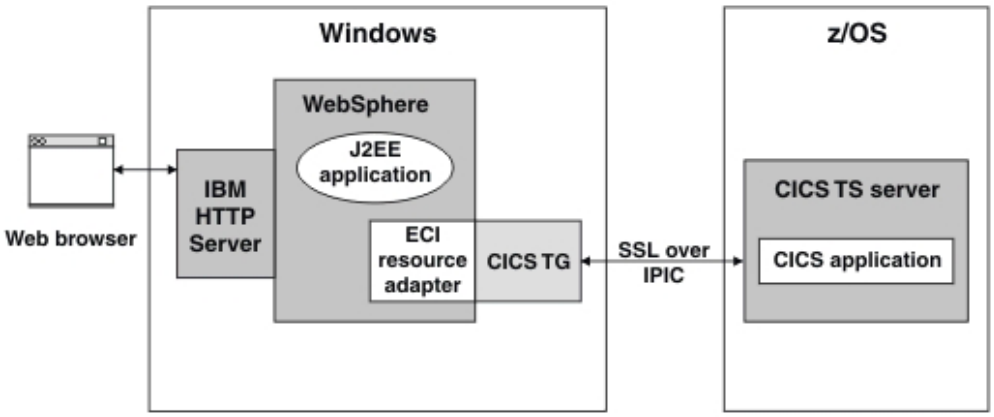
```
Program EC01 returned with data:-

        Hex: 32382f30312f31302031353a33323a34360
        ASCII text: 28/01/10 15:32:46
```

You have now completed the scenario.

Configuring SSL between CICS TG and CICS (SC07)

This scenario shows you how to configure SSL security on an IPIC connection between CICS Transaction Gateway running in local mode on WebSphere Application Server V6.1 and CICS Transaction Server V4.1.

The following figure shows the topology used in this scenario:



Follow the step-by-step instructions in this scenario using these values:

Component	Parameter	Where set	Example value
CICS server	user ID		CTGUSER
CICS server	CA certificate name	RACDCERT command	CTG CA CERT
CICS server	personal certificate name	RACDCERT command	CTG PERSONAL CERT
CICS server	keyring name	RACDCERT command	CICSSERVERKEYRING
CICS server	personal certificate file name	RACDCERT command	CTGUSER.PERSONAL.CERT
CICS server	TCPIPService	TCPIPService definition	SSL51190
CICS server	port	TCPIPService definition	51190
Java client	personal certificate file name	ikeyman	personalcert.arm
Java client	keyring file name	ikeyman	ctgclientkeyring.jks
Java client	keyring password	ikeyman	MyPassword
Java client	CTG_APPLID	WebSphere Application Server	SSLAH

Prerequisites for the SSL scenario

Before you can complete this scenario, you must ensure that the system requirements for CICS Transaction Server, CICS Transaction Gateway, and WebSphere Application Server are satisfied.

CICS Transaction Server:

- The CICS server version must be CICS Transaction Server V3.2 or later because IPIC is not available in earlier releases of CICS.
- TCP/IP services must be active in the CICS server.
  - To activate these services, set the TCP system initialization parameter to `YES`.
  - To check the status of these services, issue a CEMT INQ TCPIP command and check that the status is open.
- The SEC system initialization parameter must be set to YES to enable security.

CICS Transaction Gateway:

- CICS Transaction Gateway must be correctly installed.

WebSphere Application Server:

- WebSphere Application Server must be installed on the same machine as CICS Transaction Gateway.

To test that the scenario works successfully you can use either the supplied samples, or your own applications. If you use the supplied samples you must complete the following tasks:

- Install the sample CICS COBOL programs EC01, EC03 on the CICS server.

For information about the samples see CICS server applications.

Configuring SSL server authentication on the CICS server

To complete this task you use RACF commands to create a CA certificate, a signed personal certificate, and a keyring on the CICS server.

You perform this task by issuing ISPF RACDCERT (RACF digital certificate) commands. You use RACDCERT commands to create and maintain digital certificates, and create the keyrings that are the repositories for digital certificates.

1. Create a CA certificate that is self-signed on the server (in RACF):

   ```
   RACDCERT CERTAUTH GENCERT SUBJECTSDN(OU('CTG TEST') O('IBM')
   T('CTG CA CERT') C('GB')) KEYUSAGE(CERTSIGN) WITHLABEL('CTG CA CERT')
   ```

2. Refresh the RACF class:

   ```
   SETR RACLIST(DIGTCERT) REFRESH
   ```

3. Check that the CA certificate has been created by verifying that it exists in the output from listing the DIGTCERT class:

   a. From the ISPF main menu, enter R to display the RACF dialog.

   b. Press Enter.

   c. From the RACF - SERVICES OPTION MENU panel, enter 2 to display the RACF - GENERAL RESOURCE PROFILES panel. Press **Enter**.

   d. From the RACF - GENERAL RESOURCE PROFILE SERVICES panel, enter 8 to display the profile contents. **Press Enter**.

   e. From the RACF - GENERAL RESOURCE SERVICES - DISPLAY panel, type the class name DIGTCERT into the **CLASS** field, leaving the **Profile** field blank. Press **Enter**.

   f. From the next RACF - GENERAL RESOURCE SERVICES - DISPLAY panel, complete the following steps:

      1) Ensure that the **CLASS** field contains the class name DIGTCERT.

      2) Leave the **PROFILE** field blank.

      3) In the **DISCRETE** field, enter Yes, to select the profile type.

      4) In the **ACCESS LIST** field, enter Yes to select the access list option.

      5) Press **Enter**.

   RACF now displays a list of the selected classes; check that the list contains the DIGTCERT class that you have just created.

4. List the certificate:

   ```
   RACDCERT CERTAUTH LIST(LABEL('CTG CA CERT'))
   ```

5. Create a personal certificate on the server and sign it with your CA certificate:

   ```
   RACDCERT ID(CTGUSER) GENCERT SUBJECTSDN(OU('CTG TEST') O('IBM')
   T('CTG PERSONAL CERT') C('GB')) WITHLABEL('CTG PERSONAL CERT')
   SIGNWITH(CERTAUTH LABEL('CTG CA CERT'))
   ```

   CTGUSER must be a valid RACF user ID.

6. Refresh the RACF class:
   ```
   SETR RACLIST(DIGTCERT) REFRESH
   ```
7. Create a keyring where certificates are stored:
   ```
   RACDCERT ADDRING(CTGSERVERKEYRING) ID(CTGUSER)
   ```
8. Add the CA certificate and personal certificate to the keyring:
   a. Add the CA certificate to the keyring:
      ```
      RACDCERT ID(CTGUSER) CONNECT(CERTAUTH LABEL('CTG CA CERT')
      RING(CTGSERVERKEYRING) USAGE(CERTAUTH))
      ```
   b. Add the personal certificate to the keyring:
      ```
      RACDCERT ID(CTGUSER) CONNECT(LABEL('CTG PERSONAL CERT')
      RING(CTGSERVERKEYRING)
      DEFAULT USAGE(PERSONAL))
      ```
9. List the keyring to confirm that it contains the certificates:
   ```
   RACDCERT LISTRING(CTGSERVERKEYRING) ID(CTGUSER)
   ```

   Here is an example of the output generated by this command:
   ```
   Ring:
        >CTGSERVERKEYRING<
   Certificate Label Name               Cert Owner    USAGE       DEFAULT
   ---------------------------------    -----------   -----       -------
   CTG CA CERT                          CERTAUTH      CERTAUTH    NO
   CTG PERSONAL CERT                    ID(CTGUSER)   PERSONAL    YES
   ```
10. Export the personal certificate to a file on the server:
    ```
    RACDCERT ID(CTGUSER) EXPORT(LABEL('CTG PERSONAL CERT'))
    DSN('CTGUSER.PERSONAL.CERT') FORMAT(CERTB64)
    ```

    FORMAT(CERTB64) specifies that the certificate is stored in ASCII format.
11. Use ISPF 3.4 to view the certificate.

You have now configured SSL server authentication on the CICS server.

Configuring SSL server authentication on the client

To complete this task you use FTP to transfer the signed personal certificate from the CICS server to the client machine, then iKeyman to create a Java keystore (jks) file where the certificate is stored.

iKeyman is installed in: <install_path>/jvm160/bin

1. Transfer the personal certificate to your Client machine using an FTP client. Alternatively you can issue FTP commands on the command line.

   In "Configuring SSL server authentication on the CICS server" on page 147, you specified FORMAT(CERTB64) to ensure that the certificate was stored in ASCII. You must therefore specify ASCII when you transfer the certificate using FTP. The following example shows the FTP commands required to transfer the certificate, and the associated system responses:
   ```
   C:\ftp server
         Connected to server.company.com
         User (server.company.com:(none)): name
         331 Send password please. Password: xxx name is logged on.
         Working directory is "/u/directory".
         ftp> asc
         Representation type is Ascii NonPrint
         ftp> quote site recfm=vb
         SITE command was accepted
   ```

```
ftp> get 'CTGUSER.PERSONAL.CERT'
Port request OK. 125 Sending data set CTGUSER.PERSONAL.CERT
Transfer completed successfully.
ftp> quit
```

You have to specify the site `recfm=vb` FTP command because the server certificate is stored in a variable blocked data set.

2. Rename CTGUSER.PERSONAL.CERT to personalcert.arm.
3. Start ikeyman on your Client machine.
4. Create a new Java keystore file:
   a. From the iKeyman main menu, select **Key Database File** > **New**.
   b. From the New dialog, click the **Key database type** list then select the file type **JKS**.
   c. In the **File name** field enter the name of the Java keystore file that you want to create. In this scenario the file name is `ctgclientkeyring.jks`.
   d. Click **OK**. Because you are creating a new Java keystore file, the **Password prompt** dialog now prompts you to provide a password. Enter a password into the **Password** and **Confirm password** fields. In this scenario the password is `MyPassword`.
   e. Click **OK**.
5. Import the personal certificate personalcert.arm from the data set into the Java keystore file:
   a. Click the arrow and select **Signer certificates** from the list.
   b. Click **Add** and specify the file name and location of the file that you transferred to the client (in this scenario personalcert.arm).
   c. Click **OK**.
   d. In the **Enter a label** dialog, enter a label for the certificate. The label identifies the certificate but is not used during security processing. This scenario uses the label `cics tg racf server certificate`.
   e. Click **OK**. The server personal certificate is imported from the data set that you transferred to the client, into the Java keystore file.

You have now configured SSL server authentication on the client.

Configuring SSL client authentication

To complete this task you use iKeyman to create and export the client certificate, FTP to transfer the certificate file to the server, and a RACDCERT (RACF digital certificate) command to import the certificate into the RACF keyring.

| iKeyman is installed in: <install_path>/jvm160/bin

SSL client authentication provides extra security between the client and the CICS server. SSL client authentication builds on the security provided by SSL server authentication. SSL client authentication requires that the client keyring contains a self-signed certificate that is used to identify the connecting client.

1. Create a client certificate:
   a. Start iKeyman and open the key database file (ctgclientkeyring.jks) that you created when completing the previous task “Configuring SSL server authentication on the client” on page 148.
   b. From the menu, select **Personal Certificates**.
   c. Click **New Self-Signed**.
   d. Complete the following mandatory fields:

Key label
Enter `exampleclientcert`.

Version
Select **X509 V3**.

Key size
Select **1024**.

Common name
Specify the default value. This is the name of the machine you are using.

Validity period
Specify the default value 365 days.

e. Click **OK**.

The iKeyman tool now generates a public/private key pair.

The self-signed client certificate appears in the Personal Certificates window. The certificate has the name that you entered in the **Key label** field, in this example `exampleclientcert`.

2. Export the client signer certificate:
   a. With `exampleclientcert` highlighted, select **Extract Certificate**.
   b. On the **Data type** menu, select **Base64-encoded ASCII**.
   c. Enter the name and location of the text file containing your Client Certificate data. This scenario uses `exampleclientcert.arm`.
   d. Click **OK**.

   The exported certificate is a signer certificate generated from the personal certificate in the keyring, it does not contain the private key. Import the keyring into the keyring of all servers that need to communicate with the SSL client. The server uses the certificate to verify the identity of the client.
3. Import the client signer certificate into your RACF keyring:
   a. Transfer the file to the server into an MVS™ sequential data set using FTP, for example:

```
ftp winmvs2g
Connected to server.company.com
User (server.company.com:(none)): name
331 Send password please. Password: xxx name is logged on.
Working directory is "/u/directory".
ftp> asc
Representation type is Ascii NonPrint
ftp> quote site recfm=vb
SITE command was accepted
ftp> put exampleclientcert.arm 'CTGUSER.CLIENT.CERT.ARM'
Port request OK. 125 Sending data set 'CTGUSER.CLIENT.CERT.ARM'
Transfer completed successfully.
ftp> quit
```

   b. Add the client certificate to CLASS(DIGTCERT) using the ISPF RACF command:

```
RACDCERT ID(CTGUSER) ADD('CTGUSER.CLIENT.CERT.ARM')
WITHLABEL('CLIENT.CERT') TRUST
```

   The command returns a message confirming that the certificate has been added with TRUST status and that the class needs to be refreshed:

```
Certificate Authority not defined to RACF. Certificate added with
TRUST status
```

   c. Refresh the RACF class:

```
SETR RACLIST(DIGTCERT) REFRESH
```

d. Connect the client certificate to your RACF keyring using the ISPF RACF command:

```
RACDCERT ID(CTGUSER) CONNECT(LABEL('CLIENT.CERT')
RING(CTGSERVERKEYRING) USAGE(CERTAUTH))
```

The new signer certificate is added to the list in the Signer Certificates view, and can be used by the server to verify the identity of the client application.

You have now configured SSL client authentication.

Configuring the IPIC connection on CICS

To complete this task you use an editor to add a parameter to the startup JCL, you then edit the IPCONN autoinstall user program DFHISCIP, you then use a CEDA command to configure the TCPIPService definition and the IPCONN template definition.

1. Define the system initialization parameter for the key ring by adding the following system initialization parameter to the startup JCL:

   ```
   KEYRING=CTGSERVERKEYRING
   ```

2. Configure an IPCONN autoinstall user program DFHISCIP:
   a. Modify the sample IPCONN autoinstall program to enable the autoinstall of multiple secure IPCONNs.

      CICS provides the IPCONN autoinstall sample program DFHIS*x*IP in Assembler, C, COBOL, and PL/I , where *x* is the program language (A, D, C or P). The sample program does not use a template by default, so if you want autoinstall requests to use a template you must update the program. In this example, the COBOL user program DFHISCIP is updated.

   b. Add the following lines to DFHISCIP to ensure that, when a request arrives from a Java Client with an APPLID beginning with *SSL*, the correct IPCONN template is used to install an IPCONN with the required SSL settings. If the APPLID starts *SSLxxxxx* use the SSLIDP template.

      ```
      IF ISAIC-APPLID(1:3) = 'SSL'
             MOVE 'SSLIDP  ' TO ISAIC-TEMPLATE
             MOVE ISAIC-APPLID TO ISAIC-IPCONN
             PERFORM X000-FINIS.
      ```

   c. Compile and link-edit your program into a data set that can be picked up by your CICS server.
3. Configure a TCP/IP service:
   a. Create the following TCPIPService definition:

      ```
      CEDA  View TCpipservice( SSL51190 )
       TCpipservice   : SSL51190
       GROup          : SSLGROUP
       DEScription    : IPIC LISTENER
       Urm            : DFHISCIP
       POrtnumber     : 51190             1-65535
       STatus         : Open              Open | Closed
       PROtocol       : IPic              IIop | Http | Eci | User | IPic
       TRansaction    : CISS
       Backlog        : 00001             0-32767
       TSqprefix      :
       Host           : ANY
       (Mixed Case)   :
       Ipaddress      : ANY
       SOcketclose    : No                No | 0-240000 (HHMMSS)
       Maxdatalen     :                   3-524288
       SECURITY
       SSl            : Clientauth        Yes | No | Clientauth
       CErtificate    :
      ```

```
 (Mixed Case)
 PRIvacy        : Supported  |Notsupported | Required | Supported
 CIphers        : 0504353637383 92F303132330A1613100D0915120F0C03060201
 AUthenticate   :            | No | Basic | Certificate | AUTORegister
                             | AUTOMatic | ASserted
 Realm          :
 (Mixed Case)
 ATtachsec      :                    Local | Verify
```

b. Ensure that the SSl parameter is set to `Clientauth` so that client authentication is performed on the connection.

4. Configure an IPCONN template:

a. Create the following IPCONN definition:

```
CEDA  View Ipconn( SSLIDP   )
 Ipconn         : SSLIDP
 Group          : SSLGROUP
 DEScription    :
IPIC CONNECTION IDENTIFIERS
 APplid         : SSLIDP
 Networkid      :
 Host           :
 (Mixed Case)   :
 Port           : No                 No | 1-65535
 Tcpipservice   : SSL51190
IPIC CONNECTION PROPERTIES
 Receivecount   : 100                1-999
 SENdcount      : 000                0-999
 Queuelimit     : No                 No | 0-9999
 Maxqtime       : No                 No | 0-9999
OPERATIONAL PROPERTIES
 AUtoconnect    : No                 No | Yes
INservice       : Yes                Yes | No
SECURITY
 SSl            : Yes                No | Yes
 CErtificate    : CTG PERSONAL CERT              (Mixed Case)
 CIphers        : 0504353637383 92F303132330A1613100D0915120F0C03060201
 Linkauth       : Certuser           Secuser | Certuser
 SECurityname   :
 Userauth       : Identify     Local | Identify | Verify | Defaultuser
 IDprop         : Notallowed   Notallowed | Optional | Required
RECOVERY
 Xlnaction      : Keep               Keep | Force
```

b. Use CEDA to install the TCPIPService and the IPConn definitions.

You have now configured the IPIC connection on CICS.

Verifying the connection

To complete this task you issue a Java command then follow a series of on screen prompts.

The Java sample program EciB3 enables you to verify that the SSL connection between CICS Transaction Gateway and CICS has been correctly configured. You can optionally complete this task before completing the next task “Configuring WebSphere Application Server” on page 153.

To verify the connection:

1. Enter the following command to run the sample program EciB3. Qualify the location of the SSL key ring, for example ctgclientkeyring.jks, if required:

```
java -DCTG_APPLID=SSLAH com.ibm.ctg.samples.eci.EciB3 local:
2006 ctgclientkeyring.jks MyPassword
```

The following information is displayed on the screen:

```
CICS Transaction Gateway Basic ECI Sample 3

Usage: java com.ibm.ctg.samples.eci.EciB3 [Gateway URL]
                                          [Gateway Port Number]
                                          [SSL Keyring
                                           SSL Password]
To enable client tracing, run the sample with the following Java option:
         -Dgateway.T.trace=on

The address of the Gateway daemon has been set to local: port 2006

IPIC servers are not listed when running in local mode.
Enter URL of a CICS server, or Q to quit:
```

2. At the prompt, type the following URL: `ssl://lpar:51190`. Where *lpar* is the z/OS LPAR where CICS is running.
3. At the prompt, type a text string to send to the CICS program, for example `my test data`.
4. Type your CICS user ID: `CTGUSER`
5. Type your CICS password.

   The sample program returns verification information, for example:

```
Program EC03 returned 5 containers in channel "SAMPLECHANNEL":
        [CHAR] CICSDATETIME     = 19/05/2010 16:29:31
         [BIT] INPUTDATALENGTH  = 0000000c
        [CHAR] OUTPUTMESSAGE    = Input data was: my test data
        [CHAR] INPUTDATACCSID   = 5348
        [CHAR] INPUTDATA        = my test data
```

If the sample program returns `CICS server not found`, this indicates that the SSL connection has not been established. Check the CICS Transaction Server system log for more information, and ensure that the JKS keyring file name and password are correct (the CICS password you entered is not checked because the IPIC connection is configured with Userauth=Identify).

You have now verified the connection.

Configuring WebSphere Application Server

To complete this task you use the WebSphere Application Server Integrated Solutions Console to install the ECI resource adapter, create a connection factory, specify the connection factory properties, and deploy the ECIIVT installation verification test .ear file.

1. Install the CICS Transaction Gateway ECI resource adapter archive (RAR):
   a. In the WebSphere Administrative Console, click **Resources** > **Resource Adapters**, click **Install RAR** .
   b. From the Install RAR File window, enter the name `ECIResourceAdapter` for the RAR. Leave the class path as it is currently set, and leave the native library path blank.
   c. Click **Next** and leave the default settings.
   d. Click **OK**.
2. Create and configure a J2C connection factory:
   a. In the WebSphere Administrative Console, click **Resources** > **Resource Adapters**. Click on the ECIResourceAdapter.
   b. Click **New**.

c. Specify a name for the new J2C connection factory, for example `CF-20` and specify the JNDI lookup name `eis/CF-20`. Leave everything else with the default settings.

d. Click **OK**.

e. Click the new J2C connection factory CF-20.

f. Click **Additional Properties** >**Custom Properties**.

g. In the **Value** column of the Custom properties table, enter the values shown in the following screen:

Name	Value	Description	Required
TraceLevel	1	The level CICS Transaction Gateway diagnostic trace detail	false
TPNName	CPMI	The transaction identifier of the CICS mirror transaction	false
Password		The default password that requests through this connection use	false
UserName	CTGUSER	The default user name that requests through this connection use	false
TranName		The transaction identifier placed in EIBTRNID by CICS for the mirror transaction	false
ConnectionURL	local:	The URL of the CICS Transaction Gateway for this connection	false
ServerName	ssl://cicsserver:port	The name of the target CICS server for this connection	false
ClientSecurity		The class name of the client security exit for this connection	false
KeyRingPassword	MyPassword	The password required to access the keystore for an SSL connection	false
PortNumber	2006	The port number of the CICS Transaction Gateway for this connection	false
SocketConnectTimeout	0	The number of seconds to wait while connecting to a Gateway daemon	false
KeyRingClass	ctgclientkeyring.jks	The location of the keystore containing the certificates required for an SSL connection	false
Applid	SSLAH	The APPLID for applications using this connection	false
RequestExits		The class name of the request exits called during the execution of interactions	false
CipherSuites		The cipher suites available for an SSL connection	false
ApplidQualifier		The APPLID qualifier for applications using this connection	false
ServerSecurity		The class name of the server security exit for this connection requires the Gateway daemon to use	false

Total 17

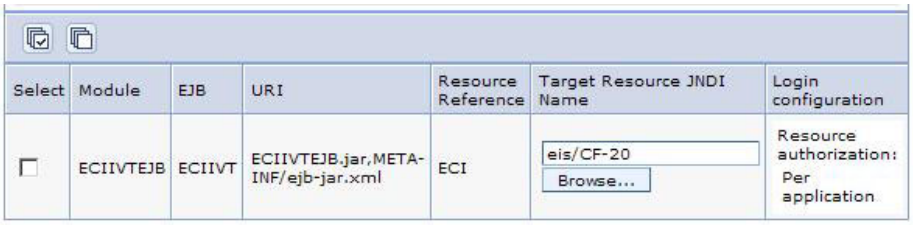
You do not have to supply a CICS password in the password field because the IPIC connection is qualified with AttachSec=Identify.

h. Save your configuration.

3. Deploy the ECIIVT ECI resource adapter installation verification test program:

a. Install the application ECIIVT.ear with a target resource JNDI name of ECIIVTBean1. The ECIIVT.ear is located within the <install_path>/ deployable directory.

b. Map the resource to your connection factory ECIIVTBean1:



c. Save your configuration.

d. Restart WebSphere Application Server if necessary (this depends on the version of WebSphere Application Server you are using).

You have now configured WebSphere Application Server.

Testing the SSL scenario

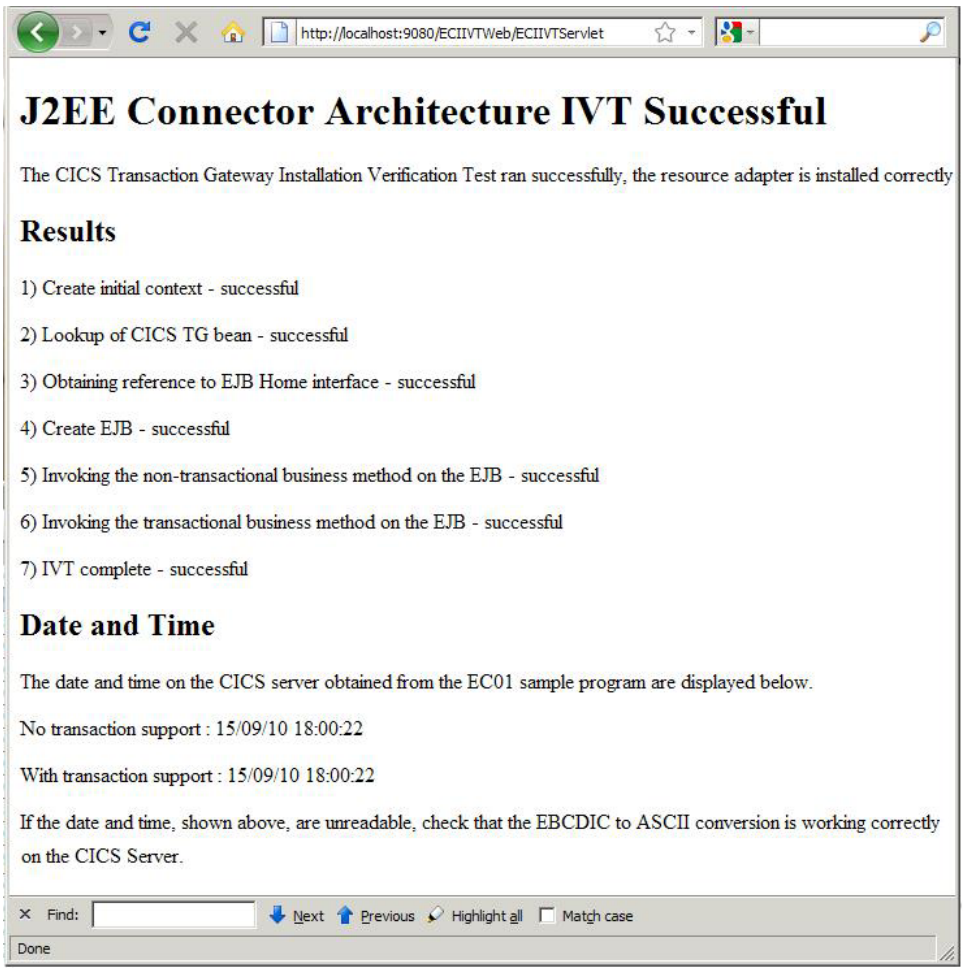
To complete this task you use a browser to go to the ECIIVT web page where you start the ECI resource adapter installation verification test.

1. Open a web browser and enter the following URL:

   ```
   http://localhost:9080/ECIIVTWeb/index.jsp
   ```

2. Click **Run IVT**.

   The J2EE Connector Architecture IVT Successful web page is displayed:



If errors have occurred, run a stack trace by clicking **Stack trace** on the IVT web page. You can also activate CICS Transaction Gateway trace in WebSphere Application Server:

a. From the WebSphere Administrative Console click **Servers** > **Application servers**.
b. Click **server1**.
c. Click **Java and Process Management** > **Process Definition** > **Java Virtual Machine**.
d. In the Generic JVM arguments pane add the following entry:
   ```
   -Dgateway.T=on
   ```
e. Restart WebSphere Application Server if necessary.
f. Look for the CICS Transaction Gateway trace in the `systemerr.log` file.

You have now completed the scenario.

Chapter 7. Security

Security mechanisms include link, bind and user security on connections, SSL client authentication, SSL server authentication, and identity propagation.

Security considerations

Authentication and authorization are performed in different locations in a CICS Transaction Gateway topology. A number of security options are available.

Authentication verifies that the user is who they say they are. Depending on topology, authentication can be based on the user ID passed with the ECI request, an SSL client certificate, or a distributed identity (identity propagation).

Authorization verifies that a user is allowed to access a particular resource for a given intent. For example to execute a method in a bean or to update a CICS resource.

Security in a local mode topology

The following figure shows the locations in a *local mode* topology where the system performs authentication and authorization. In this topology, WebSphere Application Server and CICS Transaction Gateway are both running on Windows. The EJB application in WebSphere uses the ECI resource adapter and the Client daemon to access the CICS COMMAREA application.

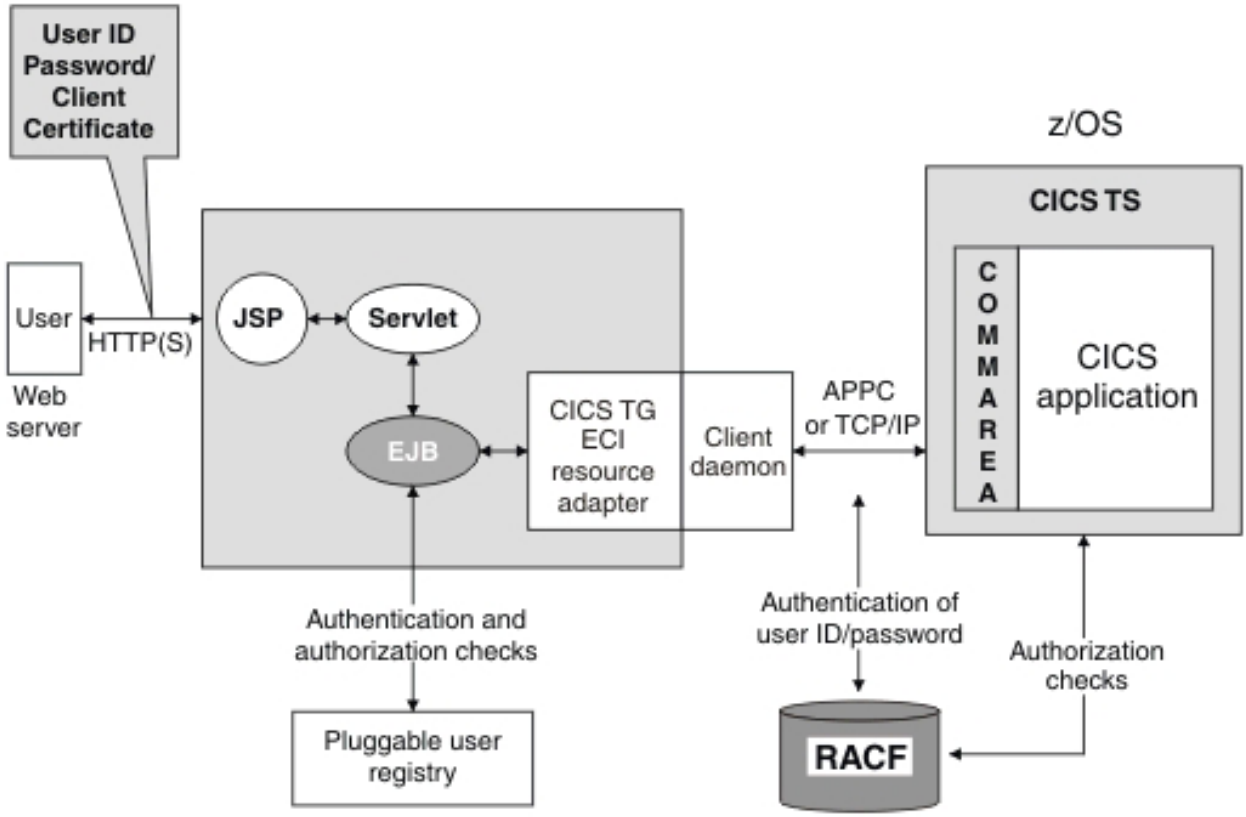


Figure 14. Security in a local mode topology

The following *authorization* options are available in this topology:

- Component-managed sign-on. With this option, security credentials are propagated to CICS by the application.
- Container-managed sign-on. With this option, security credentials are propagated to CICS by a Web or EJB container.
- Link user ID authorization checking (not available on TCP/IP connections to CICS). This provides an additional check on whether the link user ID is authorized to access the CICS resource.

The following *data integrity and confidentiality* option is available in this topology:

- HTTPS on the link between the Web server and WebSphere Application Server. The level of data encryption, server authentication and client authentication can be specified.

CICS connection security

Different security options are available on the connection when CICS Transaction Gateway is used for connecting client applications to CICS; the available options are platform and protocol dependent.

IPIC connection security

IPIC connections enforce link security to restrict the resources that can be accessed over a connection to a CICS server, bind security to prevent an unauthorized client system from connecting to CICS, and user security to restrict the CICS resources that can be accessed by a user. If the CICS server supports password phrases, a password phrase can be used for user security.

IPIC connections do not validate security credentials before sending them to CICS.

Link security

There are two ways that you can specify the link user for IPIC connections. You can use the SECURITYNAME attribute, or an SSL certificate. You can use an SSL certificate if you have a client authenticated SSL (when both the client and server have certificates). The client's certificate is mapped by RACF to a specific user ID, which is defined as the link user. This means that you can specify different link users, depending on which certificate you are using.

To specify a link user, set LINKAUTH in the IPCONN definition in CICS to one of the following settings:

1. SECUSER to use the user ID that is specified in the SECURITYNAME attribute to establish link security.
2. CERTUSER to use an SSL client certificate mapped to a user ID to establish link security.

   The IPCONN resource must refer to a TCPIPSERVICE definition that is configured for SSL and client authentication. The certificate must be mapped in RACF to your chosen user ID. For more information on certificate mapping, see the CICS Transaction Server Information Center.

Bind security

When CICS uses IPIC to communicate with a client, it uses an IPCONN resource and a TCPIPSERVICE resource. The IPCONN is used to send information to the client system's TCPIPSERVICE, which acts as a receiver. For IPIC, bind security is supported by the exchange of Secure Sockets Layer (SSL) client certificates. This allows CICS and the client system to connect successfully, and prevents an unauthorized client system from connecting.

User security

IPIC connections enforce user security to restrict the CICS resources that can be accessed by a user. The level of user security checking is specified by setting the USERAUTH attribute in the IPCONN definition in CICS. The USERAUTH setting in the IPCONN definition is comparable to the ATTACHSEC setting on other connection definitions.

- If USERAUTH=IDENTIFY is specified, a user ID that is already verified must be supplied. If the CICS TG and CICS server are not in the same sysplex, an SSL connection is required.
- If USERAUTH=VERIFY is specified, a user ID and password or password phrase must be supplied. If password phrases are used the CICS server must support password phrases.

If you are using the ECI base classes, set the user ID and password or password phrase (if required) on the ECIRequest.

To set custom properties for the ECI resource adapter set the following properties:

1. Set the flowed user name in the UserName property.
2. Set the password or password phrase (if required) in the Password property.

To override ECIConnectionSpec settings:

1. Create an ECIConnectionSpec object with the required user name and password.
2. Use this object for requests on the selected connection and in the getConnection() method of your ECI ConnectionFactory.

Identity propagation can be used as an alternative to specifying a user ID, for more information, see "Identity propagation" on page 169.

SNA connection security

SNA connections enforce link security to restrict the resources that can be accessed over a connection to a CICS server, bind security to prevent an unauthorized client system from connecting to CICS, and user security to restrict the CICS resources that can be accessed by a user.

Link security

Link security further restricts the resources a user can access, depending on the remote system from which they are accessed. The practical effect of link security is to prevent a remote user from attaching a transaction or accessing a resource for which the link userid has no authority. When link security is in use, each client is given an authority defined by a link userid. For LU6.2, all sessions in a connection can have the same link user ID, or different groups of sessions within the connection can have different link user IDs. It is also possible to specify that some groups of sessions should use link security, and that others should not.

Bind security

A security check can be applied when a request is received from, or sent to, a remote client application (when the session is bound). This is called bind-time security (or, in SNA terms, session security), and is part of the CICS implementation of the LU6.2 architecture. Its purpose is to prevent an unauthorized system from binding a session to one of your CICS systems. Bind-time security is optional in the LU6.2 architecture; you should not specify bind-time security if the remote system does not support it. SNA defines how session security is to be applied, and CICS conforms to this architecture. When connecting to a client running on another system, ensure the other system is also compatible with this architecture.

User security

In addition to the security profile set up for the link, additional restrictions can be applied to a remote client's access to the transactions, commands, and resources in CICS. For ISC over SNA and MRO links, specify the ATTACHSEC parameter on the CONNECTION definition. User security, like link security, distinguishes between transaction, resource, command, and surrogate security. User security can never increase a user's authority above that of the link.

TCP/IP connection security

TCP/IP connections enforce link security to restrict the resources that can be accessed over a connection to a CICS server, bind security to prevent an unauthorized client system from connecting to CICS, and user security to restrict the CICS resources that can be accessed by a user.

Link security

Two security protocols can be used to provide secure communication over the Internet. The first is the Secure Sockets Layer (SSL) 3.0 protocol. The second is the Transport Layer Security (TLS) 1.0 protocol, which is the latest industry standard SSL protocol and is based on SSL 3.0. The TLS 1.0 specification is documented in RFC2246. Any connections that require encryption automatically use the TLS protocol, unless the client specifically requires SSL 3.0.

Bind security

The TCPIPSERVICE resource definition specifies the security measures that are applied for each connection to CICS over TCP/IP. You can choose whether or not to use SSL, and, if you do use SSL, you choose the exact security measures that are applied; for example, the authentication method, the sending of certificates by client and server, and the encryption of messages.

User security

For ECI requests sent to CICS, you can use basic authentication to identify the user. To do this you specify ATTACHSEC(VERIFY) in the TCPIPSERVICE definition for the ECI client. Specify ATTACHSEC(LOCAL) if you do not want to identify the user.

Gateway connection security and SSL

CICS Transaction Gateway can communicate securely with CICS over a network connection using SSL (Secure Sockets Layer).

Why use SSL?

The Secure Sockets Layer (SSL) transport protocol provides authenticated, reliable, private data communications over a network connection.

Authentication

To make an environment secure, communication must be with “trusted” sites whose identities are known. SSL uses digital certificates for authentication — these are digitally signed documents which bind a public key to the identity of the private key owner.

Authentication happens at connection time, and is independent of the application or the application protocol. Authentication involves verifying that sites with which communications are established are who they claim to be. SSL authentication is performed by an exchange of certificates (blocks of data in a format described in the X.509 standard). X.509 certificates are issued and digitally signed by an external authority known as a certificate authority (CA).

Authorization

Checks are made to ensure that the authenticated users are permitted to access the system resources needed by the tasks they are performing. These resources can include computer systems, application functions, transactions, programs, databases, files, and other CICS resources.

Data integrity

Information cannot be modified during transmission.

Confidentiality

Information remains private as it passes over the connection. The information exchanged between the sender and receiver is encrypted. Only the client and the server can interpret the information.

Accountability (non-repudiation)

The sender and the receiver both agree that the information exchange took place. Accountability settles any disputes about whether or not the information was sent and received. Digital signatures ensure accountability by enabling the identification of who is responsible if something goes wrong.

What is SSL?

SSL is a security protocol that provides communications privacy. SSL enables client and server applications to communicate in a way that is designed to prevent eavesdropping, tampering, and message forgery. SSL applies only to internet protocols, and is not applicable to SNA.

How an SSL connection is established

An SSL connection is established though a handshake (a series of communications exchanges) between the client and the server.

SSL handshake

The following diagram shows what happens during an SSL handshake:

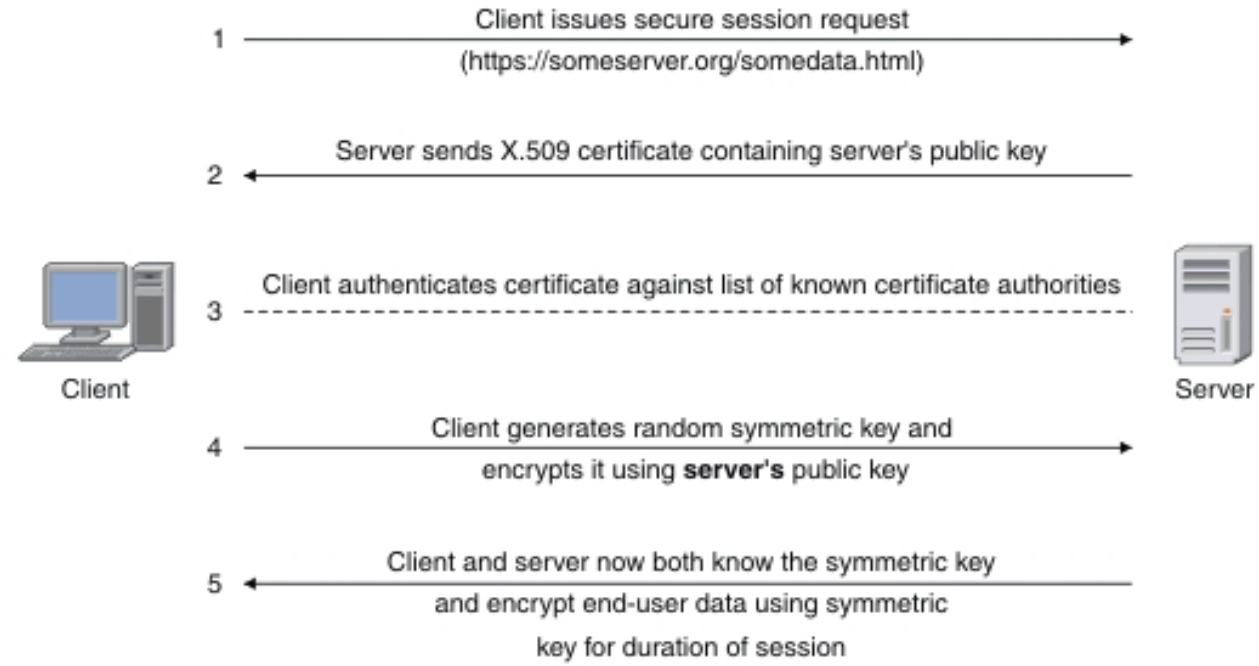


Figure 15. SSL handshake

1. The client sends a request to the server for a secure session. The server responds by sending its X.509 digital certificate to the client.
2. The client receives the server's X.509 digital certificate.
3. The client authenticates the server, using a list of known certificate authorities.
4. The client generates a random symmetric key and encrypts it using server's public key.
5. The client and server now both know the symmetric key and can use the SSL encryption process to encrypt and decrypt the information contained in the client request and the server response.

CICS Transaction Gateway supports the JSSE implementation of SSL. JSSE as supplied with the Java SDK is the only supported option. For more information, see Chapter 7, “Security,” on page 157.

Authentication

During server authentication, a connection is only established if the client trusts the server based on the information presented by the server to the client in its certificate.

During client authentication (if activated) the client sends its certificate information to the server. A connection is then only established if the client trusts the server *and* the server trusts the client, based on the information exchanged in both certificates.

Transport Layer Security (TLS):

Network connections between a JEE client and CICS can be secured by the Secure Sockets Layer (SSL) protocol, or the Transport Layer Security (TLS) protocol.

TLS is an industry-standard SSL protocol. The TLS specification is documented in RFC2246; for more information, see . http://www.rfc-editor.org/rfcsearch.html

All references to SSL in this information center also apply to TLS. Connections that require encryption automatically use the TLS protocol, unless the client specifically requests SSL. For more information on configuring CICS Transaction Gateway to use network security, see “Configuring SSL” on page 89.

No special configuration or upgrade tasks are required for using TLS, when compared with SSL.

Encryption:

Cryptography is the scientific discipline for the study and development of ciphers, in particular, encryption and decryption algorithms. These cryptographic procedures are the essential components that enable secure communication to take place across networks that are not secure. SSL encryption uses both symmetric and asymmetric keys.

Symmetric (secret) key

Secret key cryptography means that the sender and receiver share the same (symmetric) key, which is used to encrypt and decrypt the data.

The secret key encryption and decryption process is often used to provide privacy for high-volume data transmissions.

Asymmetric (public/private) key

Public/private key cryptography uses an asymmetric algorithm. The private key is known only by its owner and is never disclosed. The corresponding public key can be known by anyone. The public key is derived from the private key, but it cannot be used to deduce the private key. Either key of the pair can be used to encrypt a message, but decryption is only possible with the other key.

Digital signatures, certificates and key rings:

SSL uses digital signatures and digital certificates for establishing a trusted relationship between a sender and a receiver of information sent over a network connection.

Digital signature

A digital signature is a unique, mathematically computed, signature that demonstrates the authenticity of a transmission.

Digital certificate

A digital certificate allows unique identification. It is essentially an electronic ID card, issued by a trusted third party known as a certificate authority. Digital certificates form part of the ISO authentication framework, also known as the X.509 protocol. This framework provides for authentication across networks. A digital certificate serves two purposes: it establishes the owner's identity and it makes the owner's public key available.

A digital certificate contains the following information:

- public key of the person being certified
- name and address of the person being certified, also known as the Distinguished Name (DN)
- digital signature of the certificate authority
- issue date
- expiry date

If you send your digital certificate, containing your public key, to someone else, your private key prevents that person from misusing your digital certificate and posing as you.

A digital certificate alone is not proof of an identity; it allows verification of the owner's identity, by providing the public key needed to check the owner's digital signature. Therefore, the digital certificate owner must protect the private key that belongs with the public key in the digital certificate. If the private key is stolen, anyone could pose as the legitimate owner of the digital certificate.

Certificate authority (CA)

A digital certificate is issued by a CA and has an expiry date. When requesting a digital certificate, you supply your distinguished name. The digitally signed certificate includes your distinguished name and the distinguished name of the CA. This allows verification of the CA.

To communicate securely, the receiver must trust the CA that issued the certificate that the sender is using. Therefore, when a sender signs a message, the receiver must have the corresponding CA's signer certificate and public key designated as a trusted root key. Your Web browser has a default list of signer certificates for trusted CAs. If you want to trust certificates from another CA, you must receive a certificate from that CA and designate it as a trusted root key.

Key ring

A key ring is a file that contains the digital certificates, public keys, private keys, and trusted root keys used by a network communications security protocol such as SSL. Each certificate consists of a public key and a private key. A root certificate contains a trusted root key.

SSL requires access to key rings for the establishment of secure connections. The key rings used by the Java Secure Socket Extension (JSSE) implementation of SSL are known as *KeyStores*.

For information on how to create key rings, see “Configuring SSL” on page 89.

Cipher suites:

A cipher suite is a set of ciphers (encryption algorithms) used for encrypting sensitive information. SSL uses cipher suites to ensure security and integrity of information transmitted over a network connection. Different cipher suites provide different levels of encryption.

To allow users to select the level of security that suits their needs, and to enable communication with others who might have different needs, SSL defines cipher suites, or sets of ciphers. When an SSL connection is established, the client and server exchange information about which cipher suites they have in common. They

then communicate using the common cipher suite that offers the highest level of security. If they do not have a cipher suite in common, secure communication is not possible.

There are many different algorithms that can be used for encrypting data, and for computing the message authentication code. Some provide the highest levels of security, but require a large amount of computation for encryption and decryption; others are less secure, but provide rapid encryption and decryption. The length of the key used for encryption affects the level of security; the longer the key, the more secure the data.

The individual ciphers that can be used by CICS are dependent on the CICS Transaction Server ENCRYPTION parameter. This is a system initialization parameter which can be set for weak, medium or strong encryption.

Client security overview

CICS servers might require the Client daemon to supply a user ID and password before they permit a client connection, terminals to be installed, or transactions to be run.

This depends on the server and protocol security settings. The user ID and password are sent to the server of the transaction attach request for each conversation. A user ID and password are also required when a sign-on transaction is invoked on a sign-on capable terminal. In this instance, the user ID and password are flowed to the server as part of the 3270 data stream.

User IDs and passwords must not contain DBCS characters.

If no user ID is passed by a CICS Transaction Gateway user application, and no default is set by the CICS Transaction Gateway, the transaction is run using the mainframe CICS server's default user ID and password if the **Usedfltuser** parameter on the CICS server connection definition is set to Yes. If this parameter is set to No, security is enforced by the host CICS server and a user ID and password will need to be supplied. In each case, transactions execute in the server with the authorities assigned to the user ID authenticated.

Because the Client daemon has no security manager, it does not support user ID authentication. Configure your CICS server client connections so that incoming attach requests must specify a user ID and password. For mainframe servers, specify **AttachSec = Verify** in the CICS connection definition. **AttachSec = Identify**, which indicates that a user ID, but not password, is required, is **not** supported for client connections.

Default connection settings

The Client daemon maintains a default user ID and password for each server connection, which can be set by any of the following methods.

- CICSCLI security commands:

  ```
  cicscli -c=servername -u=userid -p=password
  ```

  The servername parameter can also be specified with the -s option.
- From C use the ESI function **CICS_SetDefaultSecurity**. This call is not available from the Java APIs.

- From C++, use the **makeSecurityDefault** method of the **CclConn** or **CclTerminal** class.

These default values are used when required on all subsequent transaction requests for that server, provided that no values have been passed on the ECI request itself, or have been set for the specific EPI terminal against which the transaction will run.

Note: If the Client daemon is running in an environment where it survives user logoff, the default user ID and password values entered by the current user are retained even when that user logs off, and are subsequently reused as required.

EPI terminal security

The Client daemon also maintains a user ID and password for each installed terminal. These values override any default values set for the server connection.

Terminal security is usually required only if using sign-on incapable terminals.

Changing the user ID and password

You can change the user ID and password at any time by following these steps.

- C programs:

  Set UserId and Password in the **CICS_EpiAttributes_t** structure on a **CICS_EpiAddExTerminal** call. Or, use the EPI function **CICS_EpiSetSecurity**. This call would typically be used to change the terminal security settings if, for example, the user's password had expired.
- C++ programs:

  Set the userid and password parameters when constructing a **CclTerminal** class object. Or, use the **alterSecurity** method of the **CclTerminal** class.
- Java Client applications:
  - EPI Request Classes

    Set the userid and password parameters when constructing an **EPIRequest** object via the **addTerminal** or **addTerminalAsync** method. Or, use the **alterSecurity** method of the **EPIRequest** class.
  - EPI Support Classes

    Create a **Terminal** object using the default Constructor, then use setUserid and setPassword to set security, or create a **Terminal** object using the extended Constructor.

Password expiry management

For CICS clients, the management of expired passwords can be handled by the ESI functions **CICS_ChangePassword** and **CICS_VerifyPassword**.

The ESI functions can be used only with CICS servers that support password expiry management (PEM). See “Supported software” on page 9 for information on supported servers. Refer to the documentation for your CICS server for information on PEM support.

To use PEM, the Client daemon must be connected to the CICS server over SNA. An External Security Manager such as Resource Access Control Facility (RACF), must also be available to the CICS server. ESI calls can be included within your ECI or EPI application. Only CICS servers returned by the **CICS_EciListSystems** and **CICS_EpiListSystems** functions are valid.

Sign-on capable and sign-on incapable terminals

Sign-on capable terminals allow sign-on transactions, either CICS-supplied (CESN) or user-written, to be run, whereas sign-on incapable terminals do not allow these transactions to be run.

If a terminal resource is installed as **sign-on capable**, the application or user is responsible for starting a sign-on transaction; the user ID and password, once entered, are embedded in the 3270 data. If the terminal resource is installed as **sign-on incapable**, the user ID and password are authenticated for each transaction started for the terminal resource.

Specifying the sign-on capability of a terminal

Terminals can be created as sign-on capable or sign-on incapable, depending both on the API function that is used to create them and the type of CICS server on which they are installed. The sign-on capability of a terminal can be specified by one of the following methods.

- C programs:

  Use **CICS_EpiAddExTerminal** and set the sign-on capability parameter in the **CICS_EpiAttributes_t** structure.
- C++ programs:

  Set the sign-on capability parameter when constructing a **CclTerminal** class object.
- Java Client applications:
  - EPI Request Classes

    Set the sign-on capability parameter when constructing an **EPIRequest** object via the **addTerminal** or **addTerminalAsync** method.
  - EPI Support Classes

    Create a **Terminal** object using the default Constructor, then use **setSignonCapability**, or create a **Terminal** object using the extended Constructor. If a terminal is in disconnected state (that is, has been disconnected, or never connected) calling **setSignonCapability** allows you to change the sign-on capability for the terminal and changes the terminal type to extended. When you connect, you connect an extended terminal with that sign-on capability. Setting the sign-on capability while a terminal is connected does not alter the connected setting; the setting is stored.

The sign-on capability of the installed terminal is returned in the terminal attributes. This will be set to SIGNON_UNKNOWN if the server does not return a sign-on capability parameter in the CTIN response.

CICS Transaction Server for z/OS:

CICS Transaction Server for z/OS supports both sign-on capable and incapable terminals, provided that they are at the prerequisite maintenance level. A terminal installation request that does not specify any sign-on capability, for example from **CICS_EpiAddTerminal**, results in a sign-on incapable terminal being installed.

For sign-on capable terminals:

- Use the **CICS_EpiAddExTerminal** call specifying a **SignonCapability** of CICS_EPI_SIGNON_CAPABLE.
- You do not need to set the userid and password fields on the **CICS_EpiAddExTerminal** call or use **CICS_EpiSetSecurity**, provided that you specify **UseDfltUser = Yes** in the CICS connection definition on the server.

- A user ID and password entered through a sign-on transaction are flowed to the server as part of the 3270 data stream and they are in a client trace.

  Specify **UseDfltUser = Yes** in the CICS CONNECTION definition, or ensure that the system administrator sets a default connection user ID and password for the client. Otherwise, the add terminal request might fail with an EPI_ERR_SECURITY return code. The default user ID must have sufficient privileges to allow the CTIN transaction to run.
- Before the user has signed on, transactions run under the default user ID for the CICS server. After sign-on, transactions run under the signed-on user ID.

For sign-on incapable terminals without terminal security:

- Use the **CICS_EpiAddTerminal** call.
- A connection user ID and password are required regardless of the setting of the **UseDfltUser** in the CICS connection definition on the server.
- Transactions run under the user ID specified in the corresponding function management header (FMH) attach request.

For sign-on incapable terminals with terminal security:

- Use the **EpiAddExTerminal** call specifying a **SignonCapability** of CICS_EPI_SIGNON_INCAPABLE.
- Set the userid and password fields on the **CICS_EpiAddExTerminal** call.
- Specify **UseDfltUser = No** in the CICS connection definition on the server to enforce security.
- Use **CICS_EpiSetSecurity** in conjunction with **CICS_VerifyPassword** and **CICS_ChangePassword** to change the security settings for an existing terminal.
- The user ID and password are flowed to the server in the FMH of the attach request and are not in a client trace.
- Transactions run under the user ID specified in the corresponding FMH attach request.

To use one of the APIs that does not support the extended EPI functionality, use CRTE through a middle tier system to get sign-on capable terminal-like functionality.

CICS Transaction Server for iSeries:

CICS Transaction Server for iSeries does not support the sign-on capability parameter in a CTIN request. A terminal installation request always results in a sign-on incapable terminal being installed.

TXSeries servers:

TXSeries servers support only sign-on capable terminals. A terminal installation request always results in a sign-on capable terminal being installed.

Identity propagation

CICS Transaction Gateway can pass user security identity information (a distributed identity) from a JEE client in WebSphere Application Server across the network to CICS Transaction Server for z/OS. The security identity of the user is preserved for use during CICS authorization and for subsequent accountability and trace purposes.

Identity propagation provides a way of authorizing requests by associating security information in WebSphere Application Server with security information in CICS Transaction Server for z/OS.

CICS Transaction Gateway supports identity propagation for JEE client requests from WebSphere Application Server to CICS Transaction Server for z/OS. Identity propagation is supported when using a CICS Transaction Gateway ECI resource adapter and an IPIC connection to CICS.

Distributed identities can be tracked using the request monitoring exits, see "Request monitoring exits" on page 257 for more information.

Benefits of using identity propagation

Identity propagation provides end-to-end security and consistent accountability, when applications in WebSphere Application Server are connected to CICS.

Identity propagation provides the following benefits:

- An end-to-end solution for security when connecting WebSphere Application Server to CICS Transaction Server for z/OS.
- A unified mechanism for authentication using security information stored in different formats on different user registries such as IBM Tivoli Directory Server or WebSphere Portal. For more information, see the documentation for WebSphere Application Server.
- "Single sign-on" authentication of users in WebSphere Application Server before they are authorized in CICS Transaction Server for z/OS.
- Consistent accountability.

Configurations that support identity propagation

A range of products and network topologies support identity propagation.

Products that support identity propagation

The following IBM products support identity propagation:

- All versions of WebSphere Application Server supported by CICS Transaction Gateway. For more information, see "Supported JEE application servers" on page 13.
- Any user registry supported by WebSphere Application Server. For more information, see the documentation for WebSphere Application Server.
- CICS Transaction Server for z/OS Version 4.1 (with APAR PK83741 and APAR PK95579), or later. For more information, see the CICS Transaction Server for z/OS information center.
- IBM z/OS Version 1.11 or later.
- IBM RACF Security Server for z/OS Version 5 or later. For more information, see *Introduction to CICS Security with RACF* in the CICS Transaction Server for z/OS information center.

Network topology for using identity propagation

Identity propagation is supported only in local mode and only on IPIC connections to CICS configured with SSL.

For more information about the topologies supported by CICS Transaction Gateway, see "Deployment topologies" on page 4.

The following example shows identity propagation in a topology with CICS Transaction Gateway in local mode:

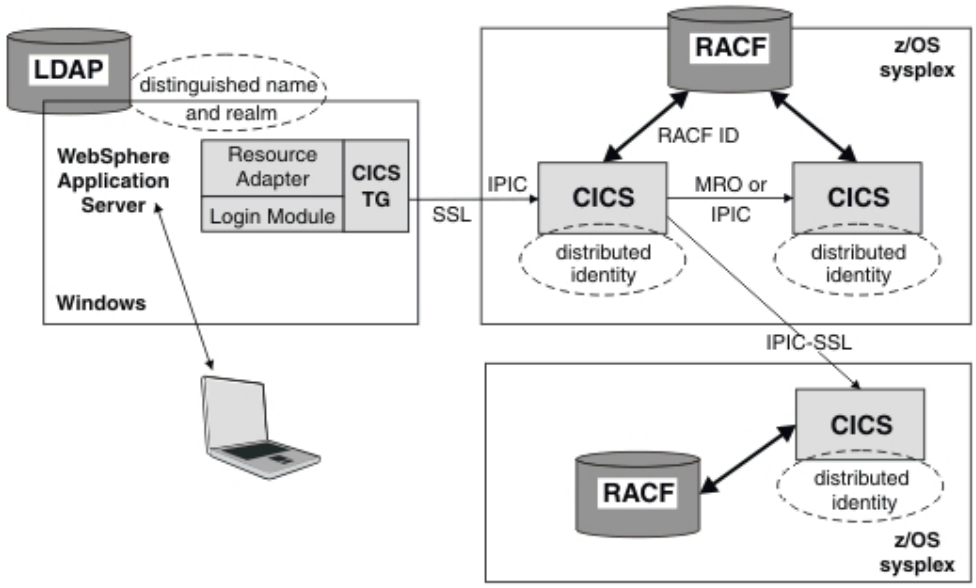


Figure 16. Example of identity propagation in a local mode topology

The user security information consists of a distinguished name and a realm name. The distinguished name uniquely identifies an entry within a user registry. The realm name represents a named collection of users and groups that can be used in a specific security context.

When the user has been authenticated in WebSphere Application Server, the security information is passed unchanged as a *distributed identity* to CICS. The distributed identity is mapped to a RACF user ID, which is used for authorization by CICS.

Precedence of distributed identities over asserted user IDs

A distributed identity takes precedence over user IDs that have been asserted directly using other mechanisms.

The identity used by CICS Transaction Server depends on whether a distributed identity has been specified and whether a valid mapping exists:

Distributed identity supplied and valid RACF mapping exists	**Distributed identity supplied but valid RACF mapping does not exist**	**Distributed identity not supplied**
The distributed identity is used and any specified user ID is ignored.	If a user ID is specified and is valid, that user ID is used.	If a user ID is specified and is valid, that user ID is used.

If a user is not authenticated by the WebSphere Application Server user registry, a distributed identity is not used even if the CICS Transaction Gateway identity propagation login module is enabled. In this situation, if a user ID has been specified in the connection factory or application, that user ID is used.

Chapter 8. Performance

The performance of individual components, including CICS Transaction Gateway, can affect overall system performance.

Related reference:

"List of statistics" on page 269
These statistics are available from the CICS Transaction Gateway.

Related information:

"Displaying statistics" on page 265
You can use the **ctgadmin** command to display statistical information about the CICS Transaction Gateway, or obtain statistics using either the C or Java Statistics API interface.

Performance indicators and factors

The performance of CICS Transaction Gateway can be measured to understand the factors that affect performance, and to use the information provided to optimize that performance.

Performance indicators

Performance indicators include the following information:

- Processor loading
- Data transfer rates
- Response times, these are useful performance indicators because they provide an understanding of which system components most affect performance.

This information helps you understand the factors that affect CICS Transaction Gateway performance and achieve the best performance from your system.

Factors that can affect performance

System components that can affect performance include:

- Web browsers
- Routers and firewalls
- Application servers (Web or JEE)
- CICS Transaction Gateway
- CICS servers

The performance of CICS Transaction Gateway also depends on whether:

- Java objects such as connections are reused
- CICS Transaction Gateway is running in local or remote mode
- Requests are synchronous or asynchronous, and whether requests are part of two-phase commit transactions
- Tracing is enabled

Factors that can improve performance

Factors that can help improve performance include:

- The multithreaded model and thread pooling to ensure the efficient reuse of connections
- Performance tuning and the use of default values to give a good balance between resource use and the ability to handle increased workload (scalability)
- Data compression can reduce the amount of data flowed over network connections. For more information see . the client and server compression sample information in the *Programming Guide*.

Monitoring performance

Ways of monitoring performance include:

- Performance monitoring tools such as RMF (Resource Management Facility), request monitoring exits, and IBM Tivoli OMEGAMON XE for CICS (on z/OS)
- Statistics for monitoring and managing system resources. For more information see "Statistics and monitoring" on page 7.

Data compression

Data compression involves encoding data so that it contains fewer bits than its non-encoded equivalent.

Data is encoded using a specific encoding algorithm. If data is sent over a network connection the sender and receiver must both understand the encoding algorithm.

Request flows

The following figures illustrate the flows that occur between the Client application and the CICS Transaction Gateway in remote and local modes. If the Client application is running under WebSphere Application Server, connection pooling can be used in remote mode to reduce the overheads associated with establishing connections.

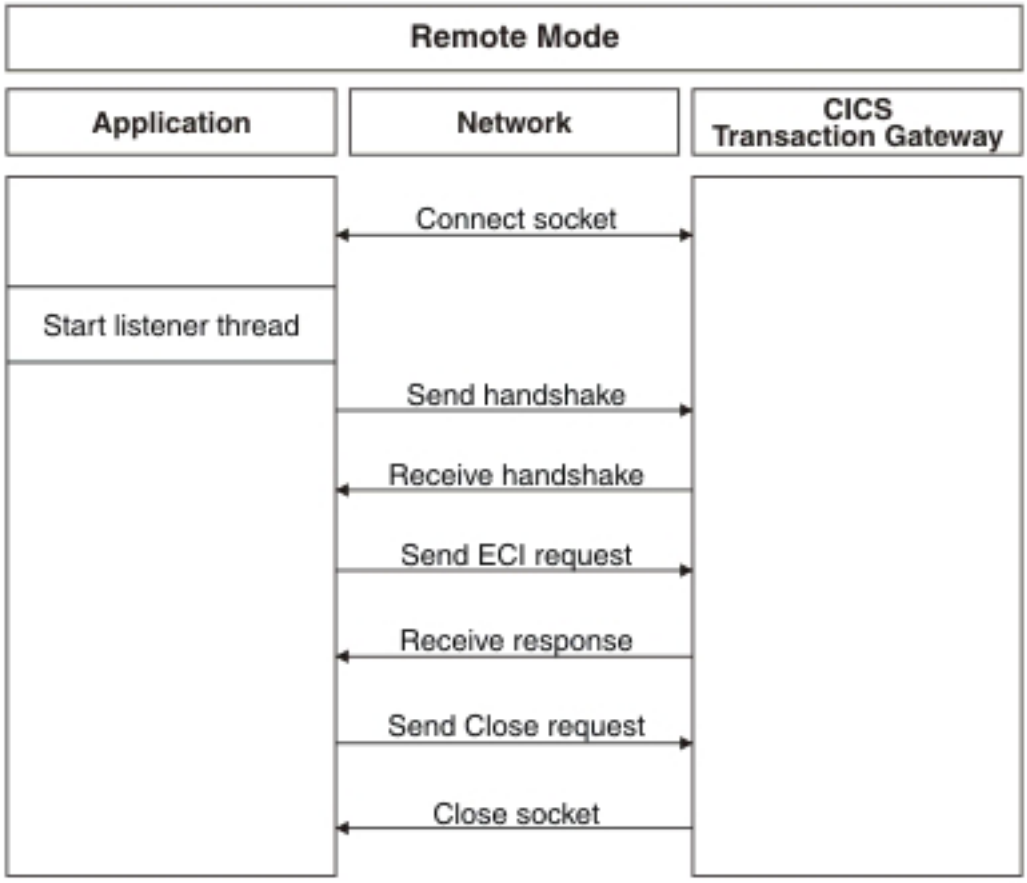


Figure 17. Request flows in remote mode

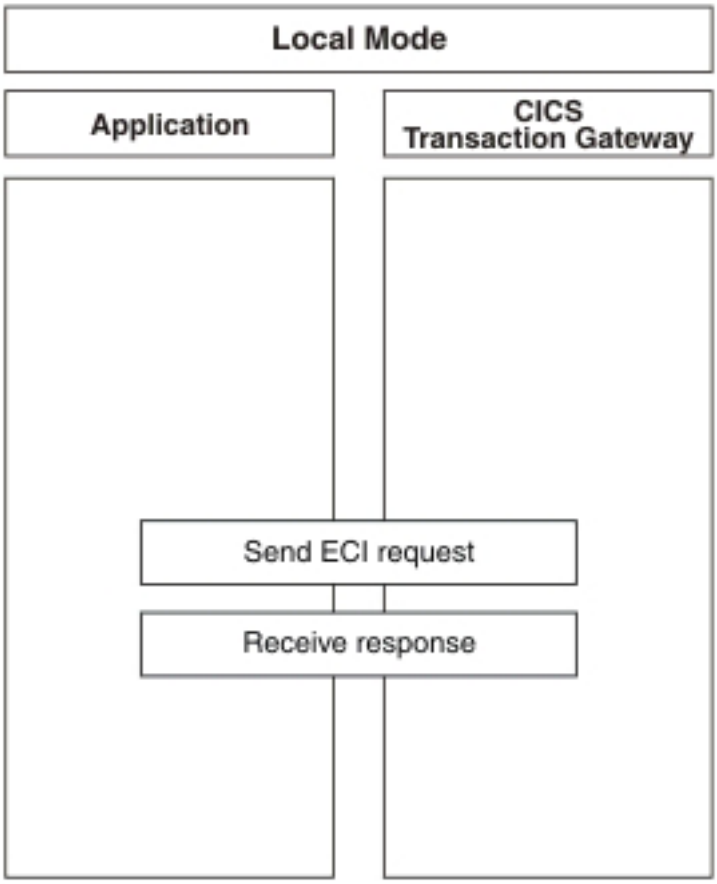


Figure 18. Request flows in local mode

Threading model

A multithreaded model provides threads that are used for handling network connections. Threads are also assigned to the requests made by remote clients and the replies received from CICS.

The threading model uses the following objects:

- Connection manager threads. These manage the connections from a particular remote Client. When it receives a request, it allocates a worker thread from a pool of available worker threads to run the request.
- Worker threads. These are allocated to run requests from remote Clients. When a worker thread has finished processing it returns to the pool of available worker threads.

You can set both the initial and maximum sizes of the resource pools for these objects; see "Configuring Gateway daemon settings" on page 70 for information on setting configuration parameters. You can also specify these limits when you start the CICS Transaction Gateway; see "Starting the Gateway daemon with override options" on page 193.

The following table shows thread limits that should be considered when setting the number of connection manager and worker threads on the various operating systems:

Table 20. Thread limits

Operating system	System-wide limit of the maximum number of threads	Process limit of the number of threads
AIX	262,143	32,768
HP-UX	No limit (30,000 kernel threads)	30,000 (refer to the max_thread_proc parameter under Configurable Kernel Parameters in the SAM utility)
Linux	Equal to the maximum number of processes	1024 (refer to your Linux Threads documentation for more information)
Solaris	No limit	No limit

The threading model is illustrated in the following figure:

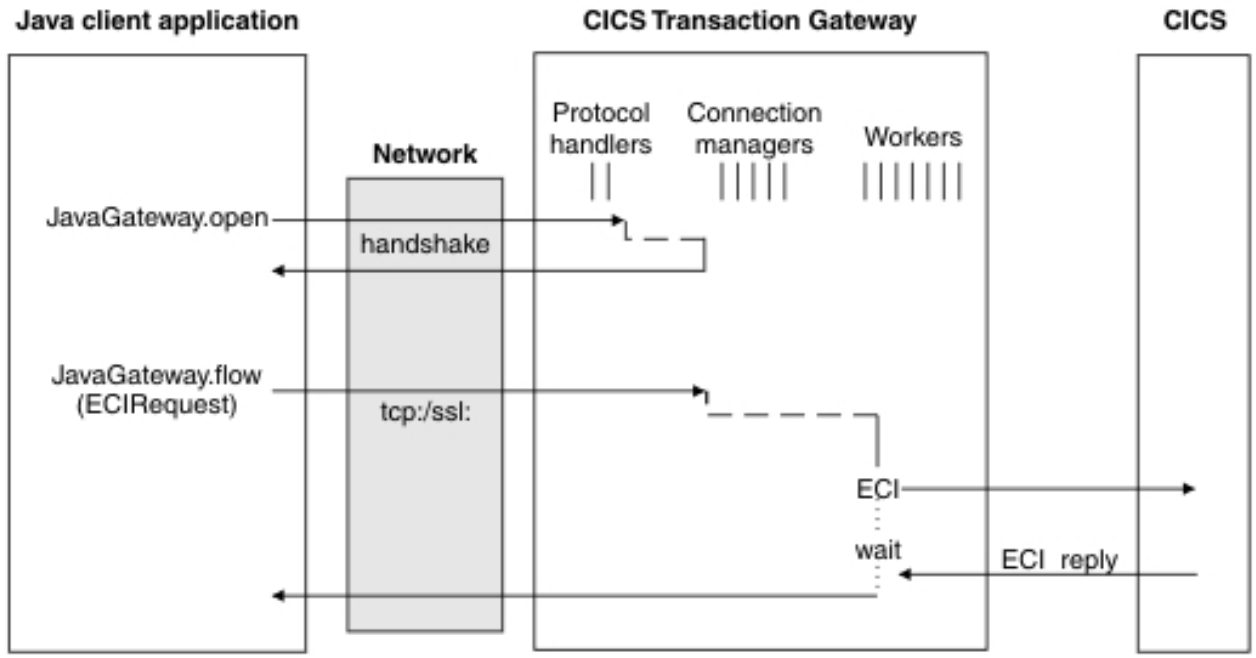


Figure 19. CICS Transaction Gateway Threading model for TCP/IP and SSL protocols using a persistent socket

Related reference:

"List of statistics" on page 269
These statistics are available from the CICS Transaction Gateway.

Related information:

"Displaying statistics" on page 265
You can use the **ctgadmin** command to display statistical information about the CICS Transaction Gateway, or obtain statistics using either the C or Java Statistics API interface.

Tuning your configuration parameters

You can tune the performance of your system by modifying values such as the number of connection manager threads and worker threads. Other values can also be modified to improve performance.

The default values that have been chosen for configuration and tuning aim to give a compromise between:

- Limiting the system resources used by CICS Transaction Gateway after it has started
- Giving the CICS Transaction Gateway the flexibility to handle increases in workload

The following factors affect performance; you might need to alter the default configuration to suit your system environment:

- Connection manager threads
- Worker threads
- Communications protocol
- Display TCP/IP host names
- Timeout values
- Connection logging settings

Connection manager threads

If the value specified for **Initial number of connection manager threads** is too high, your system will waste resources managing the threads that are not needed. See "Initial number of connection manager threads" on page 70 for more information.

If the value for **Maximum number of connection manager threads** is too low to meet all requests from applications, each new request that requires a connection manager thread must wait for a thread to become available. If the waiting time exceeds the value specified in the **Connection timeout** parameter, the CICS Transaction Gateway refuses the connection. See "Maximum number of connection manager threads" on page 70 for more information.

The design of your applications determines the number of connection manager threads you need. Incoming connections to CICS Transaction Gateway could be from a servlet, with each copy of the servlet issuing its own ECI requests, but sharing a single connection manager thread. Alternatively, the application might create a pool of connections, and ECI requests could be issued onto any connection from the pool.

CICS Transaction Gateway creates a new connection manager thread, and TCP/IP connection, each time a Java client side application creates a new JavaGateway object. This means that system performance is better if your applications issue many ECI requests using the same JavaGateway object, and from within the same thread, than if they create a new JavaGateway object for each request.

Flowing multiple requests through the same JavaGateway object also reduces the system resources required to create, and to destroy, JavaGateway objects.

Worker threads

worker threads handle outbound connections between CICS Transaction Gateway and your CICS server. The design of your applications, and the workload that you need to support, affects the number of worker threads you need: the longer your CICS transactions remain in process, the more worker threads you need to maintain a given transaction rate.

If the value specified for **Initial number of worker threads** is too high, CICS Transaction Gateway uses resources to manage threads that it does not need.

If the value is too low, CICS Transaction Gateway uses resources to search for available worker threads.

See "Initial number of worker threads" on page 71 for more information about the **Initial number of worker threads** setting.

When using ECI to call the same CICS program, you can estimate the number of worker threads you need to support a given workload by multiplying the following values:

- The number of transactions per second passing through CICS Transaction Gateway
- The average transaction response time through CICS Transaction Gateway in seconds. You can use the CS_LAVRESP statistic to calculate this response time.

Communications protocol

Your choice of protocol depends on the nature of your client applications.

Display TCP/IP host names

Selecting this option might cause severe performance reduction on some systems. See "Display TCP/IP hostnames" on page 76.

Timeout values

It is unlikely that you can improve performance by changing the default timeout values. However, you might need to change them for particular applications. See "Configuring Gateway daemon settings" on page 70 for more information on these configuration parameters.

Connection logging

The Gateway configuration setting, **Log Client connections and disconnections**, controls whether or not CICS Transaction Gateway writes a message each time that a client application program connects to, or disconnects from, the Gateway

daemon. The default is for these messages not to be written. Selecting this setting can significantly reduce performance, especially in a system where client application programs connect and disconnect frequently. See “Log Client connections and disconnections” on page 76.

Related reference:

“List of statistics” on page 269
These statistics are available from the CICS Transaction Gateway.

Related information:

“Displaying statistics” on page 265
You can use the **ctgadmin** command to display statistical information about the CICS Transaction Gateway, or obtain statistics using either the C or Java Statistics API interface.

Java considerations

Performance considerations related to Java include the size of the Java heap, whether or not the JIT (Just In Time compiler) is enabled, and whether or not Java gateway objects are reused.

Maximum heap size

If your system requires large numbers of connection manager threads you might need to increase the heap size to improve performance. How you set the heap size depends on your JVM. See the documentation for your JVM for more information.

“System environment statistics” on page 277 are available to show the following Java statistical information:

- JVM minimum and maximum heap settings
- JVM heap size after last garbage collection (GC)
- Garbage collection statistics

Just-In-Time (JIT) compiler

Use the **java -version** command to find whether or not the JIT is enabled; it is enabled by default. Immediately after a CICS Transaction Gateway has started, performance might be relatively slow because of JIT overheads. See your JVM documentation for information on JIT techniques.

JavaGateway objects

Performance is better if you flow multiple requests using the same JavaGateway object than if you create a new JavaGateway object with each request. Whenever you create and destroy a new JavaGateway object you use additional system resources for creation and destruction of the object itself, creation and destruction of any associated sockets, garbage collection.

Other system factors

A range of other system factors can affect performance.

Additional factors that can affect system performance include:

- Mode (local or remote)
- ECI COMMAREA size

- ECI containers
- CICS server transactions
- Synchronous or asynchronous ECI calls
- Extended logical units of work
- TCP/IP failure

ECI COMMAREA size

The size of the ECI COMMAREA has a large effect on performance. If you make the COMMAREA larger, you need more system resources to process it, and your response times are longer.

The **setCommareaOutboundLength** method, which is part of the **ECIRequest** object, is particularly important to performance. The amount of data that an application sends in the COMMAREA flow to CICS might be small, and the amount of data expected from CICS in return might be unknown. To improve performance significantly, and reduce network loading:

- Use the **setCommareaOutboundLength** method to ensure that you send only the required data in the outbound flow to CICS, and not the full **Commarea_Length**.

  CICS removes any null data from the COMMAREA in the return flow, and the Client daemon automatically pads out the nulls and returns the full COMMAREA to the application.
- Use the **getInboundDataLength** method to show the amount of non-null data returned.
- You can use either the setCommareaInbound method or null stripping. Use setCommareaInbound when the size of inbound data is known in advance.

See the information about ECI performance considerations when using COMMAREAs in the *CICS Transaction Gateway for Multiplatforms: Programming Guide* for more information.

ECI containers

The number and size of ECI containers have an effect on performance. As you increase the number and size of your containers, you need more system resources to process them, and your response times are longer. Load balancing can help control the flow of your data if you use large containers with multiple simultaneous requests across a single gateway.

CICS server transactions

The design of your CICS server transactions affects the performance of your system. The response time through the CICS Transaction Gateway might increase if your transactions have the following characteristics:

- Have to wait for shared resources, for example data sets or applications, to become available
- Make remote links to other CICS systems
- Are unnecessarily complex

See the performance and tuning documentation for your CICS server system for information on how to get the best performance.

Synchronous or asynchronous ECI calls

CICS Transaction Gateway has to do less processing to handle a synchronous ECI call than to handle an equivalent asynchronous call. Also, synchronous ECI calls create fewer network flows than asynchronous calls. This means that synchronous ECI calls give better performance than asynchronous calls.

Extended logical units of work

Take care when extending a logical unit of work across multiple program link calls that might span a long time period (for example, user thinking time). The logical unit of work holds various locks, and other CICS resources, on the server. This might cause delays to other users who are waiting for the same locks and resources.

Also, each logical unit of work occupies one CICS non-facility task for the duration of its execution. This means that you must define enough free tasks in the CICS server to service the maximum expected number of concurrent calls.

Performance considerations for heavy IPIC workloads

When running the Gateway daemon under load and using IPIC connected servers, you might need to increase the size of the JVM heap for the Gateway daemon if performance problems are encountered.

For information about how statistics might indicate that a JVM is short of heap storage see JVM stress causing poor performance in the Gateway daemon.

While the default maximum heap size (128MB) is adequate for an SNA or TCP/IP workload, it might be necessary to increase this to 256MB or 512MB for an IPIC workload. Ensure that the region size is increased accordingly to match any increases in heap size, otherwise the Gateway daemon might fail with a java.lang.OutOfMemory error.

IPIC provides the ability to use more than 250 worker threads in a single Gateway.

Check the value of the statistic WT_CCURR (current number of worker threads) to see if the 250 worker limit is being reached by the workload.

Performance considerations with large containers

When running with large container sizes, greater than 1 MB, it might be necessary to increase the JVM resources for the Gateway daemon and CICS.

The **-Xmx** and **-Xss** parameters might need to be changed for the Gateway daemon.

- Increase the maximum amount of heap memory available to the Gateway daemon using the **-Xmx** parameter. Failure to increase the heap could result in a JVM exception as a result of a java.lang.OutOfMemory error. Be aware that increasing the maximum heap size will reduce the amount of available process memory and therefore the number of Java threads that can be created. See "Avoiding out of memory conditions" on page 187 for more information.
- Increase the stack available to the Java threads using the **-Xss** parameter. The default value of -Xss is 256 KB with Java version 6.0. The default settings can be found in the *Java Diagnostics Guide*. Running with the a 256 KB setting for -Xss is

suitable for container sizes up to 50 MB. Failure to increase the Java thread stack size might result in a JVM exception as a result of a java.lang.StackOverflowError.

The following values might need to increase for a CICS TS server.

- MEMLIMIT - Is the limit for above-the-bar storage for the CICS server. Abend codes AITJ and APCG are an indication that MEMLIMIT might be too small.
- EDSALIM - The EDSALIM system initialization parameter specifies the upper-limit of the total amount of storage within which CICS can allocate the individual extended dynamic storage areas (EDSAs) that reside above 16 MB but below 2 GB. Abend code AIPE is an indication that the EDSALIM memory might be too small.

For more information about abend codes and their meaning, see CICS Transaction Servers.

Tracing and performance

Full tracing of CICS Transaction Gateway can degrade system performance and ideally not be used in a production environment.

Tracing the Client or the Gateway daemon reduces performance significantly. *Client trace* means all the components of the CICS Transaction Gateway that go to the Client trace file. Where possible, try to measure response times, without using tracing, through the different parts of your system, to find where delays are happening. For example, you can measure response times at the user application, or through the facilities provided by CICS and WebSphere Application Server.

The Client trace provides a greater level of control than is currently available for Gateway trace. If you need to trace the Client, take the following steps to lessen the impact on performance:

- Use memory mapped trace whenever possible; see “Memory mapped tracing” on page 247. This should be suitable in most cases, unless it was not possible to flush the buffers, for example.
- Start by specifying the API.2, CCL and DRV.1 components. If this does not isolate the problem, include extra components as necessary.

IBM does not recommend the use of the full CICS Transaction Gateway trace to monitor performance in a production environment.

Performance monitoring tools

The performance monitoring tools provide a way of measuring system performance characteristics such as transaction throughput and processor usage.

Performance monitoring tools are available on the AIX operating system. Similar tools are available on other UNIX and Linux operating systems.

Refer also to the performance and tuning documentation for WebSphere, SNA, TCP/IP, CICS Transaction Server for z/OS, and TXSeries, and the documentation supplied with your operating system.

vmstat

This tool provides statistics about virtual memory, disks, traps, and processor activity. Use this tool to determine system loading.

iostat

This tool provides processor statistics and I/O statistics for tty, disks, and CD-ROM drives.

sar (system activity report)

This tool collects, reports, or saves system activity information.

netstat

This tool shows network status.

IBM Performance Toolbox

This provides a set of useful performance tools that are integrated into a graphical user interface.Versions are available for the Solaris and HP-UX operating systems.

Statistics and performance assessment

Use statistics to assess system performance and to identify and resolve performance problems.

Indicators of performance problems are poor response time, and out of memory conditions. Statistics provide the information needed to improve system performance.

Investigating poor response times

Different system configurations and loads can lead to poor response times. Use the statistics provided by CICS Transaction Gateway to identify possible causes of poor response times and improve them.

About this task

You can use the statistics that are generated by CICS Transaction Gateway to establish the reasons why response times might be poor. It contains advice for improving poor response times:

- "Slow transaction response times in CICS" on page 185
- "Worker thread queuing in the Gateway daemon" on page 185
- "I/O errors during connection to the Gateway daemon" on page 185
- "Constraints in the network between the remote client and the Gateway daemon" on page 186
- "Constraints in the network between CICS Transaction Gateway and CICS" on page 186
- "JVM stress causing poor performance in the Gateway daemon" on page 186

Slow transaction response times in CICS
About this task

Slow transaction processing times in CICS cause increased response times.

- **Scenario:** This can occur when a CICS system becomes constrained, or when interconnected database systems cause delays in transaction processing.
- **Statistics to consider:** CS_IAVRESP, CSx_IAVRESP.

  If the value of CS_IAVRESP is higher than you anticipate for the transaction, a CICS system might be constrained, or interconnected database systems might be causing delays in transaction processing. If one CICS system in particular is experiencing slow response times, investigate the value of CSx_IAVRESP.
- **Solution:** Investigate CICS response times using CICS monitoring facilities and resolve any constraints that you find. Consider the setting of the **MAXTASK** and **TRANCLASS** CICS parameters.

Worker thread queuing in the Gateway daemon
About this task

Transaction requests queue in the Gateway daemon due to high usage of worker threads.

- **Scenario:** This can occur when the number of allocated connection managers is greater than the number of available worker threads.
- **Statistics to consider:** WT_ITIMEOUTS, WT_CCURR, WT_IALLOCHI.
  1. Have worker threads been queuing? You can establish whether this is the case by considering the value in WT_ITIMEOUTS. If:

     ```
     WT_ITIMEOUTS > 0
     ```

     worker threads have been queuing.
  2. Are all worker threads in use? If:

     ```
     WT_CCURR = WT_IALLOCHI
     ```

     all worker threads are in use.
- **Solution:** Increase the number of worker threads by amending the value of the **maxworker** configuration parameter to be equal to that of the number of connection managers. Also consider reducing the value of the **workertimeout** configuration parameter if the queuing time is unacceptably high.

I/O errors during connection to the Gateway daemon
About this task

An insufficient number of configured connection managers in the Gateway daemon causes I/O errors in a remote client.

- **Scenario:** This can occur when all available connection manager threads are allocated to remote clients.
- **Statistics to consider:** CM_IALLOCHI, CM_SMAX.

  If:

  ```
  CM_IALLOCHI = CM_SMAX
  ```

  all available connection manager threads are allocated to remote clients.
- **Solution:** Increase the maximum number of connection managers by increasing the value of the **maxconnect** configuration parameter. Consider the maximum number of worker threads that you have defined.

Constraints in the network between the remote client and the Gateway daemon

About this task

The transmission of large amounts of data causes increased response times due to high network latency over the TCP/IP connection with the Gateway daemon.

- **Scenario:** This can occur when large payloads, such as 32 KB COMMAREAs, are transmitted without the network payload being optimized using null truncation.
- **Statistics to consider:** GD_LAVRESP, GD_IAVRESP, GD_IREQDATA, GD_IRESPDATA.

  If the response time at the remote client is higher than the value reported in GD_LAVRESP or GD_IAVRESP, there might be constraints in the network between the remote client and the Gateway daemon.
- **Solution:** Select one or both of the following actions:
  - Investigate and amend the network bandwidth.
  - Modify your application design to optimize data flows by using COMMAREA null truncation, or by using the setCommareaOutboundLength() or setCommareaInboundLength() method.
  - If your application is using containers, modify the design of the application to use smaller containers.

Constraints in the network between CICS Transaction Gateway and CICS

About this task

The transmission of large amounts of data causes increased response times due to network latency over the connection between CICS Transaction Gateway and CICS.

- **Scenario:** This can occur when large payloads, such as 32 KB COMMAREAs, are transmitted without the network payload being optimized using null truncation.
- **Statistics to consider:** GD_IAVRESP, CS_IAVRESP.

  If:

  ```
  GD_IAVRESP - CS_IAVRESP = a high value
  ```

  there might be constraints in the network between CICS Transaction Gateway and CICS.
- **Solution:** Select one or more of the following actions:
  - Investigate and amend the network bandwidth.
  - Modify your application design to optimize data flows by using COMMAREA null truncation, or by using the setCommareaOutboundLength() or setCommareaInboundLength() method.
  - If you are using SNA over TCP/IP, consider a network solution that uses TCP/IP only.

JVM stress causing poor performance in the Gateway daemon

About this task

In certain circumstances, the Gateway daemon can suffer poor performance if it spends a large proportion of its time allocating storage or performing garbage collection.

- **Scenario:** This can occur if the default JVM heap size (128 MB) is used in an environment where large payloads (those greater than 16 KB) are in use and a large number of worker threads (more than 200) are in use concurrently.

- **Statistics to consider:** GD_IAVRESP, CS_IAVRESP, CM_IALLOCHI, WT_IALLOCHI, SE_CHEAPGCMIN, SE_SHEAPMAX, SE_IGCTIME, GD_IRUNTIME, SE_IGCCOUNT.
  1. Is Gateway processing time high? You can establish whether this is the case by using these statistics: GD_IAVRESP and CS_IAVRESP. If:

     ```
     GD_IAVRESP-CS_IAVRESP > 100 milliseconds
     ```

     Gateway processing time is high.
  2. Are connection managers queuing for worker threads? You can establish whether this is the case by using these statistics: CM_IALLOCHI and WT_IALLOCHI. If:

     ```
     (CM_IALLOCHI > WT_IALLOCHI) and WT_IALLOCHI > 0
     ```

     connection managers are queuing for worker threads.
  3. Is JVM garbage collection (GC) constrained? You can establish whether this is the case by using the following statistics: SE_CHEAPGCMIN, SE_SHEAPMAX, SE_IGCTIME, GD_IRUNTIME, SE_IGCCOUNT. If any of these three conditions are true the JVM GC is constrained:

     ```
     a. GC does not free at least 50% of the heap, that is
         SE_CHEAPGCMIN/SE_SHEAPMAX > 50%
     b. Time spent in GC is more than 10% of processing time, that is
         SE_IGCTIME/1000/GD_IRUNTIME > 10%
     c. Period between GC events is less than once per second, that is
         GD_IRUNTIME/SE_IGCCOUNT < 1s
     ```

     Note: SE_IGCTIME is measured in milliseconds and GD_IRUNTIME is measured in seconds.
- **Solution:** Increase Gateway daemon minimum and maximum JVM heap sizes.

Avoiding out of memory conditions

Avoid out of memory conditions by using CICS Transaction Gateway statistics.

You can use the statistics that are generated by CICS Transaction Gateway to establish whether the amount of virtual storage being used by CICS Transaction Gateway might exceed the amount that is available to the CICS Transaction Gateway address space.

- CM_CCURR
- CM_SMAX
- SE_SHEAPMAX
- WT_SMAX
- WT_CCURR

The amount of memory per process in which to run the Gateway daemon and Client daemon processes is determined by the underlying operating system. Use the tools provided by the operating system to monitor memory usage for the Client and Gateway daemons.

- Client daemon memory usage up to 100MB is generally regarded as normal. However memory usage greater than this should still be within the process limits and should not represent a problem.
- Normal usage of the Gateway daemon generally involves around 100 worker threads and 100 connection manager threads. If the total number of threads (worker and connection manager) exceeds 200, this would be regarded as high usage.

Note that, although increasing the heap size of the underlying JVM of the Gateway daemon will increase the time between garbage collection events and will generally be beneficial for performance, any improvements will gradually diminish beyond a certain size. Increasing the heap size to unnecessarily high values reduces the amount of process memory available for creating connection manager and worker threads, and increases the risk of hitting an out of memory condition within the JVM.

A high number of total threads (more than 200), or the use of IPIC connections, will result in higher demand for JVM heap allocation; in such circumstances, JVM garbage collection might become constrained. Increasing the heap size to 256 MB is adequate for most applications. See "JVM stress causing poor performance in the Gateway daemon" on page 186 for details about JVM analysis.

Chapter 9. High availability

High availability of connections to CICS servers uses dynamic server selection to allow the client application to delegate the choice of CICS server to the Gateway daemon, and to provide flexibility for deployment or change in environment.

Dynamic server selection (DSS) provides the CICS Transaction Gateway administrator with a mechanism for dynamically controlling the flow of work to CICS servers in a high availability operating environment.

A default CICS server is a simple dynamic server selection mechanism and this can be used either on its own, or in combination with another DSS mechanism.

CICS request exit

A CICS request exit program can be called by CICS Transaction Gateway at run time to dynamically select a CICS server.

A CICS request exit typically decides which CICS server to select from the CICS server name, user ID, and transaction ID passed with the request.

If the CICS request exit does not specify the name of a CICS server, CICS Transaction Gateway uses the default CICS server, if one has been defined.

If a retryable error occurs and the retry limit has not been reached, CICS Transaction Gateway calls the CICS request exit again. If the retry count limit has been reached, CICS Transaction Gateway returns the error that occurred on the last retry.

Related information:

Configuring a CICS request exit
The **cicsrequestexit** parameter specifies a class that performs dynamic CICS server selection for ECI requests and ESI requests.

Creating a CICS request exit

CICS request exit programming reference

Chapter 10. Operating

When operating CICS Transaction Gateway, it is important to start the product, and the services it uses, in the correct sequence. It is also important to shut down everything in the correct sequence, so that inflight transactions can complete.

Starting CICS Transaction Gateway

You can start CICS Transaction Gateway in console mode or as a background task.

To start CICS Transaction Gateway in console mode use the ctgstart command. For more information about ctgstart see “Starting and stopping the Gateway daemon” on page 193.

To start CICS Transaction Gateway as a background task use the ctgd command. For more information about ctgd see “Running the Gateway daemon as a background process” on page 196.

The Client daemon must start before the Gateway daemon. The ctgstart and ctgd commands both start the Client daemon, if it has not already been started, before they start the Gateway daemon.

If a local administration port has not been explicitly configured, the Gateway daemon listens for administration requests on the default port. For more information see “Port for local administration” on page 74.

Stopping CICS Transaction Gateway

When you are stopping CICS Transaction Gateway, stop the Gateway daemons in the order as follows.

1. Gateway daemon. See “Stopping the Gateway daemon” on page 195 for details.
2. Client daemon. See “Administering the Client daemon” on page 204 for details.

Shutting down the Client daemon while the Gateway daemon is still running is not supported. If the Client daemon is shut down while the Gateway daemon is still running, Client applications fail with ECI_ERR_SYSTEM_ERROR or CICS_EPI_ERR_FAILED errors, and the Client daemon is not automatically restarted. The same issue applies if the Client daemon is shut down when a long running user application is still running. To resolve the situation, shut down and restart CICS Transaction Gateway.

You can also shut down CICS transaction Gateway by issuing the ctgadmin -a shut command. For more information see “Shutting down the Gateway daemon” on page 202.

Normal shutdown

During a normal shutdown, new work is not allowed to start, and Client applications might not connect to the Gateway daemon. A normal shutdown has two phases:

1. Initiation: during this phase, the Gateway daemon waits until all work has finished, or until all Client applications are disconnected from the Gateway daemon.
2. Completion: during this phase, the Gateway daemon stops.

The following ECI requests are accepted during the initiation phase of a normal shutdown:

A Client application tries to flow an ECI request (SYNC or ASYNC) which continues a logical unit of work.

A Client application tries to flow an ECI Request (SYNC or ASYNC) which commits or backs out a logical unit of work.

A Client application tries to get a reply or wait for a reply.

The following EPI requests are accepted during the initiation phase of a normal shutdown:

A Client application tries to flow a reply to a conversational transaction.

A Client application tries to flow a request to disconnect or purge a terminal.

A Client application tries to flow a request to get event.

All other requests, or attempts to open a new connection are rejected, and an IOException is thrown.

The following EPI requests allow a normal shutdown:

ECI_GET_REPLY_WAIT

ECI_GET_SPECIFIC_REPLY_WAIT

EPI_GET_EVENT (the waitState is EPI_WAIT, for example an EPIRequest.getEvent call with the second parameter set to EPI_WAIT sets the request object to wait for events).

Other calls that are waiting to finish prevent the Gateway daemon from quiescing.

Changing the system time

The system time can be changed while CICS Transaction Gateway is running. If you do this, active processes respond to the changed time and not to the elapsed time.

When setting clocks forward or back, the behavior of timeout settings will change. When the clock is set back the elapsed time for a timeout might be increased. When the clock is set forward a timeout might expire earlier. The maximum elapsed time for a timeout will be the original timeout value plus the value of the change in time. For example, if the current time is 19:00, and a timeout is set to expire 5 minutes from now; the effect of setting the clock back by 1 hour is to increase the value of the timeout by 1 hour. The total elapsed time for the timeout is 1 hour and 5 minutes.

The following time outs are effected by this change:

- Client daemon connection
- Client daemon server retry interval
- ECI
- EPI install
- EPI read
- Gateway daemon close

- Gateway daemon idle
- Gateway daemon ping
- Server idle times
- Worker thread available thread
- Workload Manager server group

Operating the Gateway daemon

Operating the Gateway daemon involves starting and stopping this component, and performing administrative tasks such as Gateway daemon trace, dump and statistics collection.

Starting and stopping the Gateway daemon

This information describes how to start and stop the Gateway daemon.

Starting the Gateway daemon with preset options

The Gateway daemon will not start without a valid ctg.ini file.

To start the Gateway daemon with the options specified in ctg.ini, type `ctgstart` at the command prompt and press Enter.

A console session starts, and messages are displayed showing the values being used for TCP/IP.

Starting the Gateway daemon with override options

You can specify options to override parameter values specified in the configuration file, ctg.ini, or to override default JVM settings when starting CICS Transaction Gateway.

Specifying startup options

To specify one or more options at startup:

1. From a command line prompt, type **ctgstart** followed by the startup options that you require.
2. Press Enter.

A console session starts, and the system displays a series of messages that show the options currently being used for TCP/IP.

To list the startup options and their purposes, type ctgstart -?

List of options

These options are available on the ctgstart command (options are described in same order as they are listed when you type ctgstart -? c):

-port=*number*
Specifies the TCP/IP port number on which the Gateway daemon will listen.

-sslport=*number*
Specifies the TCP/IP port number on which the Gateway daemon will listen for SSL requests.

-keyring=*file*
Specifies the SSL key ring path and file name.

-keyringpw=*password*
Specifies the SSL key ring password. An error message is generated if the *keyringpw* parameter is used on its own without the corresponding *keyring* parameter in the ctgstart - command line.

-adminport=*number*
Specifies the port used to communicate with the Gateway daemon when controlling the Gateway daemon through the `ctgadmin` command.

-statsport=*number*
Specifies the TCP/IP port number on which the Gateway daemon will listen for statistics API requests.

-initconnect=*number*
Specifies an initial number of connection manager threads. See "Tuning your configuration parameters" on page 178 for performance information.

-maxconnect=*number*
Specifies a maximum number of connection manager threads. If you set this value to *-1*, no limits are applied to the number of connection manager threads. See "Tuning your configuration parameters" on page 178 for performance information.

-initworker=*number*
Specifies an initial number of worker threads. See "Tuning your configuration parameters" on page 178 for performance information.

-maxworker=*number*
Specifies a maximum number of worker threads. If you set this value to *-1*, no limits are applied to the number of connection manager threads. See "Tuning your configuration parameters" on page 178 for performance information.

-trace
Enables standard tracing (see "Tracing" on page 245). By default, the trace output shows only the first 128 bytes of any data blocks (for example the COMMAREA, or network flows). Other useful information, including the value of the CLASSPATH variable, and the code page, is shown at the start of the trace output.

Trace output is written to stderr, unless you use the -tfile option, or have used the Configuration Tool to define a default trace destination. No trace is written if the Gateway daemon does not have permission to write to the specified file. Each time the Gateway daemon is started with trace enabled, the trace file is overwritten with the new trace.

-quiet
Disables the reading of input from the console and disables writing to stdout.

-dnsnames
Enables the display of symbolic TCP/IP host names in messages. See "Display TCP/IP hostnames" on page 76 for more information.

-tfile=*pathname*
If tracing is enabled, trace output is written to the file specified in *pathname*. This option overrides the default destination for trace output (see the **-trace** option).

-x Enables full debug tracing (see "Tracing" on page 245). By default, the trace output shows the whole of any data blocks (for example the COMMAREA, or network flows). It also displays more information about the internal Gateway daemon processing than the standard trace. See the **-trace** and **-tfile** options for information on the destination for trace output.

Debug tracing significantly decreases performance.

-tfilesize=*number*
Specifies the maximum size, in kilobytes, of the trace output file.

-truncationsize=*number*
The value *number* specifies the maximum size of any data blocks that are shown in the trace. You can use this option with either the **-trace** or **-x** options to override the default size. Any positive integer is valid. If you specify a value of *0*, no data blocks are shown in the trace.

-dumpoffset=*number*
The value *number* specifies the offset from which displays of any data blocks start. If the offset is greater than the total length of data to be displayed, an offset of *0* is used.

-stack
Enables Java exception stack tracing (see "Tracing" on page 245). Java exceptions are traced, including those expected during typical operation. Expected exceptions include:

- An IOException resulting from a Java Client application ending or disconnecting from a network protocol.
- An InterruptedException resulting from a CICS Transaction Gateway protocol handler timeout such as the idle timeout, or a ping timeout.

No other trace output is created. See the **-trace** and **-tfile** options for information on the destination for trace output.

-j Passes an argument to the JVM. For example, `-j-D<name>=<value>` sets a JVM system property. See the JVM command line interpreter help for guidance in using this option. You can pass multiple arguments to the JVM. Specify the -j option multiple times to pass multiple arguments to the JVM.

-classpath=*classpath*
Specifies additional entries to append to JVM classpath that are used when launching the JVM. For example, the location of a jar file containing request exits.

-c Passes an argument to the Client daemon control program **cicscli**. For example `ctgstart -c-s -c-d -c-m=all` will call `cicscli -s -d -m=all`. For details of which options can be passed to the Client daemon, see "Administering the Client daemon" on page 204 section.

-requestExits=*exits*
List of classes to be used for request exit monitoring.

Stopping the Gateway daemon

There are two types of shutdown: normal shutdown and immediate shutdown.

During normal shutdown, the Gateway daemon waits for work in progress to complete. During this time, new work is not allowed to start. When work has completed the Gateway daemon shuts down.

During immediate shutdown, outstanding work is terminated abruptly. Existing connections are broken; requests for new connections are refused. The Gateway daemon then shuts down.

You can perform an immediate shutdown if a normal shutdown is too slow. You cannot request a normal shutdown after you have issued the command for an immediate shutdown.

To shut down the Gateway daemon:

Use the ctgadmin command. For more information see “Shutting down the Gateway daemon” on page 202. If you did not start the Gateway daemon with the -quiet option, you can stop it by typing the correct character and pressing the Enter key in the console session. Valid shutdown options are:

- Normal shutdown `Q` or `-`
- Immediate shutdown `I`

You cannot use Ctrl+C to stop the Gateway daemon.

For problems stopping the Gateway daemon, see “Gateway daemon fails to shut down” on page 231 for more information.

Running the Gateway daemon as a background process

You can run the Gateway daemon as a background process. To do this use the Gateway daemon control program <install_path>/bin/ctgd.

If the Gateway daemon runs as a background process you can complete the following tasks:

- Define the user and group that the Gateway daemon runs as.
- Configure the Gateway daemon to write information and error logs to file.
- Specify the location of the Gateway daemon configuration file (ctg.ini).
- Start the Gateway daemon with initialization parameters.
- Start and stop the Gateway daemon when the operating system starts and stops.
- Specify the TCP/IP port on which the Gateway daemon listens for local administration requests. The default port is 2810. The `ctgd` script does not use the port specified in the Gateway daemon configuration file.

The command takes start and stop parameters, and can be called during the startup and shutdown processes of your operating system.

Configuring the ctg.ini file

When the Gateway daemon is run in the background the Gateway daemon log destination must be set to file, for example:

```
log@info.dest=file
log@info.parameters=filename=/var/cicscli/cicstg.log;maxfiles=1;filesize=0;

log@error.dest=file
log@error.parameters=filename=/var/cicscli/cicstg.log;maxfiles=1;filesize=0;
```

Defining the ctgd.conf configuration file

Create a valid <install_path>/bin/ctgd.conf file. The recommended way is to copy <install_path>/samples/configuration/ctgdsamp.conf and edit the copy. You should specify the user and group that runs the Gateway daemon. The sample file contains instructions on how to create a valid configuration file.

To change the location of ctgd.conf, export environment variable $CTGDCONF use a command that has the following format:

```
export CTGDCONF=/opt/IBM/cicstg/bin/ctgd.conf
```

Starting the Gateway daemon as a background process

Use the following command:

```
ctgd start
```

The Client daemon must start before the Gateway daemon. If the Client daemon has not already been started, the ctgd command starts the Client daemon then starts the Gateway daemon.

Stopping a Gateway daemon that is already running as a background process

Use the following command:

```
ctgd stop
```

This stops the Gateway daemon immediately.

The ctgd command does not stop the Client daemon; for more information see "cicscli command reference" on page 210.

Starting/stopping the Gateway daemon with the operating system

Add a symbolic link to <install_path>/bin/ctgd in the appropriate directory, or edit /etc/inittab. For more information see your operating system documentation.

Gateway daemon administration

Gateway daemon administration tasks include starting and stopping, tracing, displaying statistics and obtaining JVM dumps.

Use the **ctgadmin** command to administer the Gateway daemon. See "ctgadmin command reference" on page 203 for more information.

You can run the ctgadmin command from a script, and check the results programmatically. See the information about return codes from the ctgadmin command in the *CICS Transaction Gateway for Multiplatforms: Programming Guide* for details of return codes used by the ctgadmin command.

Setting the Gateway trace

To set the Gateway trace issue this command at the command line.

```
ctgadmin -a trace [-tfile path] [-adminport <port>]
  [-tfilesize number] [-tlevel number]
  [[-truncationsize number][-dumpoffset number]|-fulldatadump]
```

If the default port 2810 is not used you must use the -adminport option. See Trace options for an explanation of the options.

Setting the JNI trace

To set the JNI trace issue the following command at the command line.

```
ctgadmin -a trace [-jnifile path] [-jnilevel number] [-adminport <port>]
```

If the default port 2810 is not used you must use the `-adminport` option. For more information see "Trace options" on page 198.

Querying trace settings

To query trace settings issue this command at the command line.

```
ctgadmin -a trace [-adminport <port>]
```

If the default port 2810 is not used you must use the -adminport option. See Trace options for an explanation of the options.

Trace options

These options are available for use with the `ctgadmin -a trace` command.

To get help on these options, issue the following command:

```
ctgadmin -a trace -?
```

Option	Short form	Comments
-dumpoffset	-of	Specifies the offset from which displays of any data blocks start, for example `512`. If the offset is greater than the total length of data to be displayed, an offset of 0 is used. This option applies only to the Gateway trace, not JNI trace. You cannot use this together with the `fulldatadump` option.
-fulldatadump	-fd	Sets the `dumpoffset` to 0 and ignores any value specified in `truncationsize`. This option applies only to the Gateway trace, not JNI trace.
-jnifile	-jf	Specifies the name of the output file for JNI tracing. You must specify a value for this option. If you do not, an error is displayed. JNI trace is output as plain text, and there is no requirement to use a particular extension for the file name.
-jnilevel	-jl	**0** Off. No trace information is output. **1** On.
-tfilesize	-ts	Specifies the maximum size, in kilobytes, of the Gateway trace output file, for example `50000`.
-tfile	-tf	Specifies the output file for Gateway tracing, for example ./tracefile.trc. If you do not specify a value for this option, trace output is sent to the console.

Option	Short form	Comments
-tlevel	-tl	Specifies the Gateway trace level. Permitted values are: **0** Off. No trace information is output. **1** Exception tracing. Only exceptions are traced. This is equivalent to `ctgstart -stack`. See"Starting the Gateway daemon with override options" on page 193. **2** Trace exceptions, and entry and exit of methods. **3** Trace exceptions, some internals, and entry and exit of methods (equivalent to `ctgstart -trace`). **4** Full debug tracing (all trace points), equivalent to `ctgstart -x`).
-truncationsize	-tr	Specifies the byte at which to stop the hex dump, for example `2000`. It defines the end point, not the number of bytes to display. So if on a dump of size 40 you set the `dumpoffset` to 11, and the `truncationsize` to 25, you will see 15 bytes (from 11 to 25). You cannot use this together with the `fulldatadump` option. This option applies only to the Gateway trace, not JNI trace.

Dumping diagnostic information

Dumps contain diagnostic information that can be used when investigating system problems. Various options are available when obtaining dumps.

Dump options are available for use with the `ctgadmin -a dump` command.

To get help on the available dump options, issue the following command:

```
ctgadmin -a dump -?
```

If the IBM JVM is used, a subset of the options can be used to provide JVM dumps. The IBM JVM can produce a Java heap dump, a Java dump, or a Java system dump. These are produced by a running JVM, and can be requested during typical operation of the CICS Transaction Gateway. The dumps contain diagnostic information that can be used when investigating system problems.

For further information on IBM JVM dumps, see the *IBM Java Diagnostic Guide* at IBM Java Diagnostic Guide.

Parameters

There are no short forms of the parameter names.

Parameter	Comments
-all	Generates all dumps. This option must be specified as the only option and cannot be combined with other dump options.
-ctginfo	Generates a dump containing information about the configuration of CICS Transaction Gateway.
-heap	(IBM JVM only) Generates a Java heap dump.
-java	(IBM JVM only) Generates a Java dump.
-jvm	Generates a dump containing current JVM memory usage.
-jvmstack	Generates a dump containing only the Java call stack.
-system	(IBM JVM only) Generates a Java system dump.

Responses

The Gateway daemon responds to a dump request with a message to the console.

IBM JVM dump responses

Messages from the JVM contain the name of the dump, and indicate whether the dump was successful. The JVM messages are sent to the Gateway error log.

Possible responses to the dump request are as follows:

The Gateway daemon successfully processed the dump request; the dump request completed successfully.

Null response received from the Gateway daemon during the dump request; the Gateway daemon received the dump request, but returned an invalid or null response.

The dump type is unsupported in the remote JVM; the remote JVM does not support the requested dump type.

An invalid response was returned from the Gateway daemon during the dump request; the Gateway daemon received the dump request, but the response returned was invalid.

The Gateway daemon encountered a serious error while processing the dump type; the Gateway daemon received the dump request, but an error was detected.

Some dump types are unsupported in the remote JVM; the remote JVM executing the Gateway daemon does not support some dump types.

Querying statistics

Options are available for selectively querying statistics.

To query all statistics, issue this command at the command line:

```
ctgadmin -a stats -gs
```

To get help on these options, issue the following command:

```
ctgadmin -a stats -?
```

Option	Short form	Comments
-getstats	-gs	Lists all available statistics.
-getstats=*query string*	-gs=*query string*	Lists statistics for the IDs specified in *query string*.
-resourcegroups	-rg	Lists available resource group IDs
-statids	-si	Lists available statistical IDs
-statids=*resource group ID*	-si=*resource group ID*	Lists available statistical IDs for the specified resource group, or list of resource groups.
-stattype=*statistics type*	-st=*statistics type*	Lists available statistical values for the specified statistics types.

Related information:

"Displaying statistics" on page 265
You can use the **ctgadmin** command to display statistical information about the CICS Transaction Gateway, or obtain statistics using either the C or Java Statistics API interface.

Request monitoring exit control

Options available for commands sent to all configured and active request monitoring user exits.

To get help on the available rmexit options, issue the following command:

```
ctgadmin -a rmexit -?
```

Option	Short form	Comments
-command=*command*	-cmd=*command*	The command that will be sent to all configured and active request monitoring user exits. This is a string. The eventFired() method is driven with a RequestEvent command. The *command* input data will be included as a string in the data map with RequestData key "CommandData".

CICS request exit control

Options available for commands sent to the configured CICS request exit.

To get help on the available options, issue the following command:

```
ctgadmin -a crexit -?
```

Option	Short form	Comments
-command=*command*	-cmd=*command*	The command that is sent to the configured CICS request exit. This is a string. The eventFired() method is driven with a RequestEvent command. The *command* input data is included as a string in the data map with RequestData key "CommandData".

Shutting down the Gateway daemon

The shutdown command and available options.

To shut down the Gateway daemon issue this command at the command line:

```
ctgadmin -a shut [-immediate] [-adminport <port>]
```

If the default port 2810 is not used you must use the -adminport option.

Options are available for use with the `ctgadmin -a shut` command.To get help on these options, issue the following command:

```
ctgadmin -a shut -?
```

Option	Short form	Comments
-immediate	-imm	Specifies that the Gateway daemon shuts down immediately. If you do not specify this option, a normal shutdown is performed.

Viewing message help

Help is available for CICS Transaction Gateway messages by using an option on the **ctgadmin** command.

Usage:

To obtain help for a message, enter the `ctgadmin` command followed by the `-msg` option, then the message number:

```
ctgadmin -msg nnn[I|E|W]
```

Where *nnn* is the message number, and`[I|E|W]` are optional identifiers that denote the message severity (information, error, or warning).

Here are some examples:

```
ctgadmin -msg CTG8897I
ctgadmin -msg CCL9483I
ctgadmin -msg CCL9483
ctgadmin -msg 8897I
ctgadmin -msg 8897
ctgadmin -msg 816
```

To obtain help for the `-msg` option itself:

```
ctgadmin -msg -?
```

Sample output

Help for error and warning messages is detailed and includes the following information:

- message number
- message text
- explanation
- system action
- user response

For example:

```
CTG8929W Environment variable AUTH_USERID_PASSWORD is not set;
user ID and password authentication is not enabled

Explanation: The environment variable AUTH_USERID_PASSWORD controls whether
or not user IDs and passwords supplied on CICS requests are authenticated.

System action: User ID and password authentication is not enabled.

User response: Set the environment variable AUTH_USERID_PASSWORD to
either 'YES' or 'NO'. Set this environment variable to YES to enable user ID
and password authentication. Set it to NO to disable user ID and password
authentication.

CTG8218I The command completed successfully
```

Help for information messages does not have to contain the same level of detailed and includes the following information:

- message number
- additional information

For example:

```
CTG6524I Successfully started handler for the 'protocol' protocol
on port 'port'

This is an information message.

CTG8218I The command completed successfully
```

For more information see the messages section of the *CICS Transaction Gateway Programming Reference*.

Getting help

To get general help or help on a particular action issue one of these commands at the command line.

Issue the following command for general help:

```
ctgadmin -?
```

To get help on a particular action, issue a command like the following:

```
ctgadmin -a action -?
```

ctgadmin command reference

Options that can be used with the **ctgadmin** command. Options are not case-sensitive.

```
ctgadmin [-adminport <number>] [-dbg [<filename>]]
         -a <action> [<action specific options>]
```

The options are:

Option	Description
-a dump	"Dumping diagnostic information" on page 199
-a trace [-tfile path] [-tfilesize number] [-tlevel number]	"Setting the Gateway trace" on page 197
-a trace [-jnifile path] [-jnilevel number]	"Setting the JNI trace" on page 197
-a trace	"Trace options" on page 198
-a trace -?	"Querying trace settings" on page 198

Option	Description
-a stats	"Querying statistics" on page 200
-a crexit	"CICS request exit control" on page 201
-a rmexit	"Request monitoring exit control" on page 201
-a shut	"Shutting down the Gateway daemon" on page 202
-msg	"Viewing message help" on page 202
-?	"Getting help" on page 203
-adminport	The port used to communicate with the IBM CICS Transaction Gateway service (Windows) or the Gateway daemon (UNIX). The default is 2810.
-dbg	Output trace to stderr, or <filename> if specified.

Operating the Client daemon

Operating the Client daemon requires you to complete administrative tasks such as setting override options, trace characteristics, and security, also starting and stopping this component.

Administering the Client daemon

Use the cicscli command and associated command options to start and stop communication with CICS servers, to check the availability of CICS servers, and to perform other tasks related to Client daemon administration.

The Client daemon starts automatically when you start the Gateway daemon.If you stop the Gateway daemon, this action does not stop the Client daemon; you must shut down the Client daemon after stopping the Gateway daemon. For more information see "Stopping CICS Transaction Gateway" on page 191.

You cannot use the cicscli command to list or work with IPIC connections to CICS.

Follow the links to see examples of how to use the cicscli command. For information on the cicscli command syntax see "cicscli command reference" on page 210.

Starting the Client daemon

You can issue a command to start the Client daemon or use the Start menu.

To start the Client daemon, enter:

```
cicscli -s
```

To start the Client daemon and start communication with a CICS server, enter:

```
cicscli -s=servername
```

where *servername* is the name of a CICS server.

Starting server connections

You can start a connection to a CICS server when the Client daemon is already running.

Enter the following command:

```
cicscli -s=servername
```

where *servername* is the name of a CICS server.

If you change and reinstall the CICS connection definition, you must stop and restart the connection.

You do not have to start server connections explicitly. When a request is sent to a server, the Client daemon automatically starts the server connection if it is not already established.

Shutting down the Client daemon

You can shut down the Client daemon for all connected servers, after all outstanding units of work have completed or without completing outstanding units of work, and shut down the session with a particular server.

To shut down the Client daemon for all connected servers, after all outstanding units of work have completed, enter:

```
cicscli -x
```

To shut down the session with a particular server after all outstanding units of work have completed, enter:

```
cicscli -x=servername
```

where *servername* is the name of a CICS server. This stops only the session with the named server; it does not shut down the Client daemon or connections to other servers.

To shut down the Client daemon for all connected servers, without completing outstanding units of work, enter:

```
cicscli -i
```

To shut down the session with a particular server without completing outstanding units of work, enter:

```
cicscli -i=servername
```

where *servername* is the name of a CICS server. This stops only the session with the named server; it does not shut down the Client daemon or connections to other servers.

If the Client daemon does not shut down completely, this might be because the Client daemon process, **cclclnt**, remains active. To stop this process, enter the command

```
kill -2 pid
```

where *pid* is the numeric process id of **cclclnt**.

Do not use the `kill -9` command as this stops a process without allowing its resources to be released; those resources remain active until you restart the system.

Restarting the Client daemon normally

You can shut down the Client daemon for all connected servers, after all outstanding units of work have completed, and then start it again.

To shut down the Client daemon for all connected servers, after all outstanding units of work have completed, and then start it again, enter:

```
cicscli -y
```

`cicscli -y` is equivalent to `cicscli -x` followed by `cicscli -s`. This does not re-establish server connections or trace settings.

Restarting the Client daemon immediately

You can shut down the Client daemon for all connected servers, without completing outstanding units of work, and then start it again.

To shut down the Client daemon for all connected servers, without completing outstanding units of work, and then start it again, enter:

```
cicscli -j
```

`cicscli -j` is equivalent to `cicscli -i` followed by cicscli -s. This does not re-establish server connections or trace settings.

Starting client tracing

You can trace the Client daemon from the startup sequence and improver performance while tracing by using memory mapped tracing.

To start tracing the Client daemon, enter:

```
cicscli -d[=size]
```

Where `size` is an optional parameter which specifies the maximum size of data, in bytes, to be traced with any individual trace message. The default is 512 bytes.

To trace the Client daemon from the startup sequence, enter the -s and -d options together:

```
cicscli -d -s
```

The Client daemon writes trace entries to the cicscli.bin file in the /var/cicscli subdirectory. New trace entries overwrite any existing entries in the trace file. If required, make a backup copy of the old trace file before you start tracing.

Performance while tracing is on can be improved by using *memory mapped tracing*. With memory mapped tracing, data is stored initially in memory, and flushed to disk by the operating system's paging mechanism. For more information, see “Memory mapped tracing” on page 247. For important security information, see “Security considerations for trace and log files” on page 208.

To use memory mapped tracing, do the following:

1. Turn on wrapping trace by setting the **Client trace file wrap size (KB)** field in the Configuration Tool to a value greater than 0; see “Client trace file wrap size (KB)” on page 111 for details of the field setting.
2. When you turn on tracing, specify the `-b` option as well as the other options, for example:

   ```
   cicscli -d -b
   ```

   or

   ```
   cicscli -d -m=component_list -b
   ```

Use the cicsftrc utility to format the trace file; see “Formatting the binary trace file” on page 248.

Specifying the trace components

Use the cicscli command plus options to specify which client components to trace.

The format of the command is:

```
cicscli -m[components]
```

where [components] is a comma-separated list of component identifiers for the components to be traced.

For example:

```
cicscli -m=TRN,API.2
```

where TRN, API.2 specifies that tracing is performed on the internal interprocess transport between Client processes, and on the client API layer levels 1 and 2.

The following component identifiers are available for use with the cicscli -m command:

Identifier	Trace
ALL	All components. Use this option if performance allows, and consider using the binary formatting tool to filter information. For more information see "Formatting the binary trace file" on page 248.
API	Synonymous with API.1.
API.1	The client API layer level 1. This traces the boundary between the Client application and the Client daemon.
API.2	The client API layer level 1 and 2. This gives level 1 plus additional parameter tracing.
CCL	The Client daemon.
CLI	The cicscli command interface.
CPP	The C++ class libraries.
DEF	The default components, that is the API, CCL, DRV and TRN components.
DRV	Synonymous with DRV.1.
DRV.1	Protocol driver tracing level 1. This traces data sent and received and provides supplementary information about failures.
DRV.2	Protocol driver tracing level 2. This traces internal flows through the protocol drivers and interactions with other software components.
EMU	The cicsterm and cicsprnt emulators.
LMG	The Workload Manager
TRN	Synonymous with TRN.1.
TRN.1	Internal interprocess transport between Client processes level 1.
TRN.2	Internal interprocess transport between Client processes level 2. Use if entries in the Client log refer to functions such as FaarqStart, FaarqStop, FaarqGetMsg or FaarqPutMsg.

The -m option specifies the components to trace. To turn tracing on use the -d option. Consider using wrapping trace (-b option) for improved performance:

```
cicscli -m=component_list -d -b
```

If you specify -m with no parameters, a list of the possible component identifiers is displayed, with an 'x' for each component that it is currently enabled for tracing.

You can also specify settings for trace components using the Configuration Tool. For more information see “Configuring trace settings” on page 109. Component tracing specified using cicscli overrides component tracing specified using the Configuration Tool. If component tracing is not specified either by the cicscli command or by using the Configuration Tool, these default components are traced: API, CCL, DRV and TRN.

Stopping client tracing

You can enter a command to stop tracing the Client daemon or, stop tracing automatically.

To stop tracing the Client daemon, enter:

```
cicscli -o
```

Trace stops automatically if you enter the `cicscli -x command.`

Security considerations for trace and log files

The Client daemon restricts access to the client trace and log files. By default, these are typical files, not links, called cicscli.bin, and cicscli.log, in the /var/cicscli subdirectory. You can specify names for these files using the Configuration Tool.

The trace file:

Normally, the file permissions on cicscli.bin allow only the owner, and group to write to the file, and only the owner to read it. However, a user who has write access to the /var/cicscli subdirectory can delete cicscli.bin regardless of the file permissions.

If however you use the `-b` option (see “Starting client tracing” on page 206), every process is given read access to the trace files.

If you do not want unauthorized users to have access to the cicscli.bin file, do not give them write access to the /var/cicscli subdirectory. For example, a command such as:

```
chmod 755 /var/cicscli
```

allows users to see files in /var/cicscli subdirectory but not to create, delete, or move them.

The Client daemon prevents you from starting tracing if an unauthorized user has deleted and recreated cicscli.bin.

Security and the error and warning log file:

The file permissions on cicscli.log allow only the owner (root) and group to read and write the file.

To improve security further:

- Set the permissions on the /var/cicscli subdirectory to restrict general access:

  ```
  chmod 0711 /var/cicscli
  ```

  This means users cannot even see which files are in this directory.
- Allow ECI and EPI programs, and terminals, to start the Client daemon, but allow only the root user to perform all other client administration. To do this,

restrict access to the <install_path>/cicscli binary file, and allow general read and execute access to the /var/cicscli subdirectory.

Setting security for server connections

You can set a default user ID and password on a server connection.

You can issue the following commands when the Client daemon is running:

To set a user ID and password to use when accessing server *servername*, enter:

```
cicscli -c=servername -u=userid -p=password
```

To specify security parameters when you start a connection to a server, enter:

```
cicscli -s=servername -u=userid -p=password
```

-c=servername
identifies the name of the server to which security information in the form of a user ID and password is to be associated.

-u=userid
sets the default user ID to be used when accessing the server specified by the -c or -s option. Specifying -u or -u= (that is, no user ID is specified) resets the associated user ID to a null value.

-p=password
sets the default password to be used when accessing the server specified by the -c or -s option. Specifying -p or -p= (that is, no password is specified) resets the associated password to a null value.

For ECI applications, any user ID and password specified in the ECI parameter block override values set by the cicscli command.

Displaying the version of CICS Transaction Gateway

You can display information about the version and build level of CICS Transaction Gateway.

To display this information enter the command:

```
cicscli -v
```

Controlling cicscli command messages

You can disable the display of all messages produced by the command.

For example, enter:

```
cicscli -q
```

A -w prompt to press the Enter key is issued before the command completes, this enables the user to confirm that they have read the information and error messages displayed on the screen.

Listing the connected servers

You can list all connected servers being used by the Client daemon, and their status.

To list all connected servers enter:

```
cicscli -l
```

The status of connected servers is updated as a result of requests being flowed and protocol specific events. The status returned is the last known state of connected servers, which might not be the same as the current state.

Destination for error messages

By default, the Client daemon sends error messages to the log file cicscli.log in the /var/cicscli subdirectory.

On AIX, you can redirect these messages to another target device or file by using the **swcons** command. You cannot use the `-e` option to send messages to the console.

cicscli command reference

Options that can be used with the cicscli command.

- All client control commands have options that are identified by a leading dash - character.
- On the Linux and Solaris platforms you can replace the dash - character by a forward slash / character for all options.
- On the AIX and HP-UX platforms you can replace the dash - character by a forward slash / character for all options except the `?` option, where you must use a dash - character.

All options of the form *-x=variable* can contain spaces in the variable part, if the variable part is enclosed in double quotes. Double quotes within variables must be entered as `\"` , that is with a backslash preceding the double quote.

```
cicscli [[-s=servername|[-x[=servername]]|[-i[=servername]]|[-j]|[-y] ]
[-l]
[[-d[=nnn] [-b] [-m[=components]]]|-o]
[-c=servername [-u[=userid]] [-p[=password]]]
[-w|-q]
[-v]
[-y|-j]
```

Enter `cicscli -?` to display help for the cicscli command.

The options are:

Option	Description
-b	“Starting client tracing” on page 206
-c=servername	“Setting security for server connections” on page 209
-d=[nnn]	“Starting client tracing” on page 206
-i[=servername]	“Shutting down the Client daemon” on page 205
-j	Shuts down the Client daemon immediately and then restarts it. A restart involves shutting down the Client daemon, waiting for it to shut down, and then starting it again. cicscli -j is equivalent to cicscli -i followed by cicscli -s. Server connections are not re-established when the Client daemon is restarted.
-l	“Listing the connected servers” on page 209
[-m [components]]	“Specifying the trace components” on page 207
-o	“Stopping client tracing” on page 208
-p=password	“Setting security for server connections” on page 209

Option	Description
-q	“Controlling cicscli command messages” on page 209
-s[=servername]	“Starting server connections” on page 204
-u=userid	“Setting security for server connections” on page 209
-v	“Displaying the version of CICS Transaction Gateway” on page 209
-w	“Controlling cicscli command messages” on page 209
-x[=servername]	“Shutting down the Client daemon” on page 205
-y	Restarts the Client daemon normally. A restart involves shutting down the Client daemon, waiting for it to shut down, and then starting it up again. cicscli -y is equivalent to cicscli -x followed by cicscli -s. Server connections are not reestablished when the Client daemon is restarted. Using -y is the preferred way of restarting the Client daemon.

Chapter 11. 3270 terminal emulation and printing

The CICS Transaction Gateway 3270 terminal emulator enables CICS applications to send output to an attached printer.

3270 terminal emulation

The 3270 terminal emulator included with CICS Transaction Gateway can be used for running CICS applications without the need for a separate 3270 emulator product. The 3270 terminal emulator enables multiple CICS 3270 emulation sessions to run with one or more CICS servers. The screen color attributes and keyboard settings of the client terminal emulator can be customized with mapping files. This is useful for example if keyboard layouts are required to comply with a company standard. CICS Client terminal definitions can be automatically installed on the CICS server and do not have to be predefined.

3270 printer support

The 3270 printer support included with CICS Transaction Gateway enables a printer terminal to be defined on CICS Transaction Gateway so that CICS applications running on a server can send output to an attached printer. Output can be sent to a printer attached to a port such as LPT1, or a command can be issued to process the data into a format more suitable for specialized printers. CICS 3270 printer support uses CICS 3270 terminal emulation.

cicsterm emulator

cicsterm is a CICS 3270 terminal emulator. You can have multiple terminal emulators running simultaneously.

Using cicsterm

The cicsterm command enables you to work with terminal emulators and terminal emulator sessions.

You use the cicsterm command to:

- Start a emulator session.
- Specify the characteristics of a terminal emulator.
- Specify the file name of a keyboard mapping file. The default cicsterm keyboard mappings are those defined by cicskey.ini. For information about customizing keyboard mappings see “Keyboard mapping for cicsterm” on page 106.
- Specify the file name of a color mapping file. The default the cicsterm color mappings are those defined by cicscol.ini. For information about customizing color mappings see “Customizing the screen colors for cicsterm” on page 108.
- Run multiple terminal emulation sessions at the same time.

You cannot connect to CICS over IPIC using the cicsterm command because the CICS servers are not listed.

cicsterm options

A number of options and parameters are available when using the **cicsterm** command to start a 3270 terminal emulator.

The options are as follows:

-a Specifies that the terminal emulator is *not* sign-on capable. By default, cicsterm creates terminal emulators that are sign-on capable.

For more information on sign-on capability, see the information about specifying terminal sign-on capability in the *CICS Transaction Gateway for Multiplatforms: Programming Guide*.

-c=*colorfile*
Identifies the name of a color mapping file to be used with the emulator; see "Customizing the screen colors for cicsterm" on page 108 for more details. If you omit this parameter, the environment variable CICSCOL is assumed to identify the color mapping file. If CICSCOL is not defined, a file name of cicscol.ini in the <install_path>/bin subdirectory is assumed.

If the parameter is specified as -c=, that is, the color mapping file name is omitted, the emulator runs without any color definitions.

-e Specifies that when extended attributes are included in a 3270 datastream received by **cicsterm** and no color attribute is set, the color of a field is determined by both the intensity and protection attributes. The colors used to display the field are then determined by the **normal_unprotected**, **intensified_unprotected**, **normal_protected** and **intensified_protected** definitions in the color mapping file. By default, when this option is not specified, the 3270 datastream contains extended attributes and no color attribute is set, the color of fields is determined by the **default** and **default_highlight** definitions in the color mapping file. See "Customizing the screen colors for cicsterm" on page 108 for further information on color mapping.

-f=*printfile*
Specifies the name of a file to which the output of print requests is appended. If you do not specify a full path, *printfile* is created in the <install_path>/bin directory. If the name of the file contains embedded blanks, it must be surrounded by double quotes ("). Any double quotes within the name of the file must be entered as backslash double quote (\").

If neither the -f or -p parameters is specified, the **Print command** or **Print file** configuration settings define the command, file, or default action to take with print requests.

-k=*keyfile*
Identifies the name of a keyboard mapping file to be used with the emulator; see "Keyboard mapping for cicsterm" on page 106 for more details. If this parameter is omitted, the environment variable CICSKEY is assumed to identify the key mapping file. If CICSKEY is not defined, a file name of cicskey.ini in the <install_path>/bin subdirectory is assumed.

-m=*modelname*
Specifies the name of a Model terminal definition, as known at the server to which the emulator is to connect, to be used to define the terminal characteristics. If neither this parameter nor -n=netname is specified, any Model terminal definition value from the configuration file is used. If no Model terminal definition value has been specified in the configuration file, the server's default terminal definition is assumed.

If the parameter is specified as -m= (that is, the modelname is omitted), any Model terminal definition value specified in the configuration file is ignored, and the server's default terminal definition is assumed.

This option is case-sensitive.

-n=*netname*
Specifies the name of a particular terminal definition at the server that this emulator is to be installed as. The precise interpretation of netname varies between servers.

This option is case-sensitive.

-p=*printcmd*
Specifies an operating system command used to process the temporary print file generated when print requests are received by the terminal emulator. If the command contains embedded blanks, enclose it in double quotes (“). Any double quotes within the command must be entered as backslash double quote (\“).

The temporary print file is post-processed by appending the file name to the command, and running that command. Print output can then be copied to a local printer, copied into a permanent file, processed further for inclusion into a document, and other similar actions. If the temporary file is to be processed by a print command, the command is responsible for deleting the temporary file.

If neither the -f or -p parameters is specified, the **Print command** or **Print file** configuration settings define the command, file, or default action to take with print requests.

-q Disables the display of all messages output by the command.

-s=*servername* **or -r=***servername*
Specifies the name of the server that the terminal emulator is to be connected to. This server name must correspond to an entry in the configuration file.

You can specify -s, or -r, but not both. If neither parameter is specified, the first server entry in the configuration file is used.

If the parameter is specified as -s or -r (that is, no server name is provided), and the configuration file identifies more than one potential server to which the Client daemon can connect, the user is prompted to select from a list of available servers. These prompts are generated even if messages have been disabled (the -q parameter is specified).

If there is only one potential server identified in the configuration file, that server is used and the user is not prompted.

-t=*initialtransid*
Identifies the initial transaction to be invoked for this terminal. If this option is omitted, any initial transaction specified in the configuration file is run. The string can be up to 128 characters long, specifying both a transaction name, and parameters to be passed to the transaction. The transaction name is the first four characters or the characters up to the first blank in the string. The rest of the string is the parameter data.

If the parameter is specified as -t= (that is, the initialtransid is omitted), any initial transaction specified in the configuration file is ignored.

This option is case-sensitive.

Ensure that transaction that you specify here does not require terminal input to complete.

-w Prompts the user to press the Enter key, to confirm that messages output to the screen (both informational and error) have been read, before the command completes.

-? Causes the parameter syntax to be listed; any other options specified are ignored.

Stopping a terminal emulator

To stop a terminal emulator, enter the string specified by the Terminal exit configuration setting from an empty terminal screen, and then press Enter. The default string is `EXIT`

cicsterm and user exits

You can use cicsterm to drive EPI user exits.

The EPI user exits, and how cicsterm can use them, are described in the information about EPI user exits in the *CICS Transaction Gateway for Multiplatforms: Programming Guide*.

cicsterm and RETURN TRANSID IMMEDIATE

When an application running from a cicsterm session issues one of the following commands the transaction named in the TRANSID option starts immediately without any user input.

```
EXEC CICS RETURN TRANSID(name) IMMEDIATE
EXEC CICS RETURN TRANSID(name) IMMEDIATE INPUTMESSAGE(data-area)
```

When the INPUTMESSAGE option is specified, the contents of the *data-area* are passed to the new transaction, and the screen is not updated with the *data-area* contents.

Issuing these EXEC CICS commands from cicsterm does not result in the StartTranExit or ReplyExit user exits being driven. See the information about EPI user exits in the *CICS Transaction Gateway for Multiplatforms: Programming Guide* for more information.

Using clients for X-Window System

There are some known problems with certain clients for X-Window System (such as Exceed).

These include corruption of the text on the title bar of the window that you are trying to display, for example, with the Configuration Tool. In some cases the title bar might be missing. The problems are with the client used, not the CICS Transaction Gateway.

There are two ways to solve this problem:

1. Start a window manager before launching the Configuration Tool.
   The window manager HWM is shipped with Exceed.
2. Alter the Exceed configuration:
   a. Launch Xconfig
   b. Select **screen definition**

c. Set the Window Manger to *native*
d. Make sure **Use Native WM for Embedded Clients** is checked

If you use this second way you cannot run any other window manager.

Keyboard mapping in cicsterm

Keyboard mapping in cicsterm is governed by the terminal type that you are using.

Many terminal types do not support all of the function keys that can be used in a CICS application. If cicsterm does not recognize some of the key mappings defined by the <install_path>/bin/cicskey.ini file, try using a different terminal type. For example, on AIX systems use *aixterm* in preference to *xterm*.

cicsterm restrictions

There are some restrictions when you use the cicsterm command.

- 3270 field outlining is not supported.
- Window resizing is not supported. If you resize the window in which cicsterm is running the display becomes distorted.
- CICS Transaction Server interprets cicsterm as a remote terminal. The execution diagnostic facilities CEDF and CEDX have some restrictions on remote terminals. For more information see your CICS Transaction Server documentation.
- The maximum supported screen size for cicsterm terminal emulators is 27 rows by 132 columns. This is because the Client daemon uses the 12-bit addressing ASCII-7 subset of the 3270 data stream architecture.

cicsterm command reference

The tasks and associated parameters that the **cicsterm** command supports.

```
cicsterm [-s=servername|-r[=servername]]
[-t=[initialtransid]]
[-k=keyfile]
[-c=colorfile]
[-m=modelname]
[-n=netname]
[-a]
[-p=printcmd|-f=printfile]
[-q|-w]
[-?]
```

You can replace the dash (-) with the forward slash (/) character.

For example:

- On Linux and Solaris you can replace the dash by a forward slash (/) for all parameters.
- On AIX and HP-UX you can replace the dash by a forward slash (/) for all parameters except the **?** parameter, where you must use a dash sign.

The following table shows the tasks and associated parameters that the **cicsterm** command supports:

Option	Description
-a	Specify a terminal emulator that is sign-on incapable

Option	Description
-c	Specify the name of the color mapping file
-e	Specify when extended attributes are included
-f	Specify a file to which print files are appended
-k	Specify the name of the keyboard mapping file
-m and -n	Define the 3270 terminal emulator characteristics
-p	Determine the print file processing
-q	Inhibit all output messages
-r and -s	Start a 3270 terminal emulator
-t	Specify the initial transaction
-w	Wait for confirmation before completing

Issue a single cicsterm command, with all the parameters you require, as shown in the following example:

```
cicsterm -s=CICSTSW -t=CESN -k=mykeys.ini -c=mycols.ini
         -n=cicsv123 -f=clprint.txt -q
```

In this example:

-s=CICSTSW
A 3270 terminal emulator is started for the server `CICSTSW`.

-t=CESN
The initial transaction is `CESN`.

-k=mykeys.ini
The keyboard mapping file is `mykeys.ini`.

-c=mycols.ini
The color mapping file is `mycols.ini`.

-n=cicsv123
The 3270 terminal emulator characteristics are defined by the terminal definition `cicsv123`.

-f=clprint.txt
The print file will be appended to the file `clprint.txt`.

-q The display of messages output by the command is disabled.

All cicsterm parameters are optional. If you enter the cicsterm command without any parameters, defaults are taken from the configuration file.

cicsprnt emulator

cicsprnt is a CICS 3270 printer terminal emulator. You can have multiple printer terminal emulators running simultaneously.

Using cicsprnt

An application running on a server can direct output to a printer in one of these ways.

An application running on a server can direct output to a printer in one of the following ways:

- An application running from a terminal can initiate printing by sending a map or data with the PRINT indicator set.
- A user can start a 3270 Print Terminal Emulator, at a client, using the cicsprnt command. A 3270 Print Terminal Emulator must be started for a netname or model terminal definition predefined in the server's terminal tables. Output is directed to such a device by starting a transaction against the printer device.
- At client workstations you can use the PrintScreen key, as defined by the keyboard mapping file. Blank lines are not printed by default. A blank line is defined as a line that contains only null characters or non-displayable fields, or is undefined in the BMS map. However, there are options available to override the default behaviour of cicsprnt, see "cicsprnt options" for more information.

cicsprnt options

A number of options and parameters are available when using the **cicsprnt** command to start a 3270 printer terminal emulator.

The dash (-) can be replaced with the forward slash (/) character.

- On Linux and Solaris, you can replace the dash by a forward slash (/) for all parameters.
- On AIX and HP-UX, you can replace the dash by a forward slash (/) for all parameters except **?** for which you must use a dash.

For example:

```
cicsprnt -m=modelname|-n=netname
[-s=servername|-r[=servername]]
[-t=[initialtransid]]
[-p=printcmd|-f=printfile]
[-q|-w]
[-j]
[-z]
[-b]
[-?]
```

The options are:

-b By default, **cicsprnt** does not print blank lines. A blank line is defined as a line that contains only null characters or non-displayable fields, or is undefined in the BMS map. This option causes blank lines in the data stream to be printed.

-f=*printfile*
Specifies the name of a file to which the output of print requests is appended. If you do not specify a full path, *printfile* is created in the <install_path>/bin directory. If the name of the file contains embedded blanks, it must be surrounded by double quotes ("). Any double quotes within the name of the file must be entered as backslash double quote (\").

If neither of the -f or -p parameters is provided, the **Print command** or **Print file** setting in the configuration file defines the command, file, or default action to take with print requests.

-j Specifies that cicsprnt should concatenate all EXEC CICS SEND PRINT commands issued on a server transaction into a single print job. This print job is issued when the transaction terminates. Otherwise cicsprnt generates a separate print job for every EXEC CICS SEND PRINT command issued for a server transaction.

-m=*modelname*
Specifies the name of a model terminal definition, as known at the server to which the 3270 Print Terminal emulator is to connect, to be used to define the terminal characteristics. If this parameter is not specified, any Model terminal definition value from the configuration file is used. If no Model terminal definition value has been specified in the configuration file, the server's default terminal definition is assumed.

You must specify either the -m or the -n option, or both.

This option is case-sensitive

-n=*netname*
Specifies the name of a particular terminal definition at the server that this 3270 Print Terminal emulator is to be installed as. The precise interpretation of netname varies between servers. For example, on TXSeries for AIX it is a netname.

You must specify either the -m or the -n option, or both.

This option is case-sensitive.

-p=*printcmd*
Specifies a command used to process the temporary print file generated when print requests are received by the terminal emulator.

If the command contains embedded blanks, the command must be surrounded by double quotes (“). Any double quotes within the command must be entered as backslash double quote (\“).

If neither of the -f or -p parameters is specified, the **Print command** or **Print file** setting in the configuration file defines the command, file, or default action to take with print requests.

The temporary print file is post-processed by appending the file name to the command, and executing the resultant command, so reports can be copied to a local printer, copied into a permanent file, processed further for inclusion into a document, and so on. If the temporary file is processed by a print command, the command is responsible for deleting the temporary file.

-q Disables the display of all messages output by the command.

-s=*servername* **or -r=***servername*
Specifies the name of the server that the printer is to be connected to. This servername must correspond to an entry in the configuration file. You can specify -s, or -r, but not both.

If neither parameter is specified, the first server entry in the configuration file is used.

If the parameter is specified as -s or -r (that is, no servername is provided) then, if the configuration file identifies more than one potential server to

which the Client daemon can connect, the user is prompted to select from a list of available servers. These prompts are generated even if the -q parameter is specified.

If there is only one potential server identified in the configuration file that server is used and the user is not prompted.

-t=*initialtransid*
Identifies the initial transaction to be invoked for this printer. If this option is omitted, any initial transaction specified in the configuration file is run. The string can be up to 128 characters long, specifying both a transaction name, and parameters to be passed to the transaction. The transaction name is the first four characters or the characters up to the first blank in the string. The rest of the string is the parameter data.

If the parameter is specified as -t= (that is, the initialtransid is omitted), any initial transaction specified in the configuration file is ignored.

Note: Be careful that transactions that you specify either here or in the configuration file do not require terminal input to complete.

This option is case-sensitive.

-w Prompts the user, before the command completes, to press the Enter key, to confirm that messages output to the screen (both informational and error) have been read.

-z When cicsprnt is running in a DBCS locale and a field containing mixed DBCS and SBCS character is displayed, blank spaces are inserted between DBCS and SBCS characters. This option suppresses the insertion of spaces.

-? Causes the parameter syntax to be listed; any other options specified are ignored.

cicsprnt and user exits

You can use cicsprnt to drive EPI user exits.

The EPI user exits, and how cicsprnt can use them, are described in the information about EPI user exits in the *CICS Transaction Gateway for Multiplatforms: Programming Guide*.

cicsprnt and RETURN TRANSID IMMEDIATE

cicsprnt does not support these commands, unlike cicsterm.

```
EXEC CICS RETURN TRANSID(name) IMMEDIATE
EXEC CICS RETURN TRANSID(name) IMMEDIATE INPUTMESSAGE(data-area)
```

For more information, refer to “cicsterm and RETURN TRANSID IMMEDIATE” on page 216.

cicsprnt restrictions

There are some restrictions when you use the cicsprnt command.

- If the system running the Client daemon supports DBCS, it is assumed that the printer attached to the processor also supports DBCS. Conversely, if the system does not support DBCS, the Client daemon does not send DBCS data to the printer.

cicsprnt command reference

The tasks and associated parameters that the **cicsprnt** command supports.

```
cicsprnt [-s=servername|-r[=servername]]
[-t=[initialtransid]]
[-m=modelname]
[-n=netname]
[-p=printcmd|-f=printfile]
[-q|-w]
[-j]
[-z]
[-b]
[-?]
```

The dash (-) can be replaced with the forward slash (/) character.

For example:

- On Linux and Solaris you can replace the dash by a forward slash (/) for all parameters.
- On AIX and HP-UX you can replace the dash by a forward slash (/) for all parameters except the **?** parameter, where you must use a dash sign.

The following table shows the tasks and associated parameters that the **cicsprnt** command supports:

Option	Description
-s and -r	Start a 3270 print terminal emulator.
-t	Specify the initial transaction.
-n and -m	Define the 3270 printer terminal emulator characteristics.
-p	Determine the print file processing.
-f	Specify a file to which print files are appended. The default location is <install_path>/bin.
-w	Wait for confirmation before completing.
-q	Inhibit all output messages.
-j	Issue one print job per transaction.
-z	Suppress the insertion of a blank character between single-byte and double-byte characters.
-b	Causes blank lines in the data stream to be printed.
-?	Display help

Issue the cicsprnt command once with all the parameters you require, as shown in this example:

```
cicsprnt -s=CICSTSW -n=P123 -t=XPRT -f=clprint.txt -q
```

In this example:

-s=CICSTSW
A 3270 print terminal emulator is started for the server CICSTSW.

-n=P123
The 3270 print terminal emulator characteristics are defined by the terminal definition `P123`

-t=XPRT
The initial transaction is `XPRT`.

-f=clprint.txt
The print file to which print requests are appended is `clprint.txt`, in the default location (<install_path>`/bin`).

-q The display of messages output by the command is disabled.

You must specify at least one of these parameters:

-n=*netname*

-m=*modelname*

All other parameters are optional, and defaults are taken from the configuration file. Full details of the parameters are given in "cicsprnt command reference" on page 222.

Chapter 12. Resolving problems

If a problem occurs you should first do some preliminary checks to try and narrow down the cause. You can then try and analyze the problem in more detail using tools such as trace, debug, or diagnostic commands. A wide range of additional resources are also available for problem solving, including: forums and newsgroups, IBM Technotes, IBM Developerworks, and IBM Redbooks. You can also contact your IBM Support organization.

Introduction to problem determination

The usual procedure is to start with a symptom, or set of symptoms, and trace back to the cause.

Sometimes, you cannot solve the problem yourself if, for example, it is caused by limitations in the hardware or software you are using.

If the cause of the problem is CICS Transaction Gateway, contact IBM, as described on the support page at: `http://www-01.ibm.com/software/htp/cics/ctg/support/`

Preliminary checks

Before you examine the cause of the problem in more detail, perform these preliminary checks. These might highlight a simple cause or, at least, narrow the range of possible causes.

As you go through the questions, make a note of anything that might be relevant to the problem. Even if the observations you record do not at first suggest a cause, they could be useful to you later if you need to carry out systematic problem determination.

Has the system run successfully before?

If the system has not run successfully before, it might not have been installed or configured correctly. You can check that CICS Transaction Gateway installed correctly by running one of the sample programs; for more information, see “Using the sample programs to check your configuration” on page 123. You can also use the “JCA resource adapter installation verification test (IVT)” on page 121 to test that the connection from WebSphere Application Server through CICS Transaction Gateway to CICS Transaction Server is working correctly.

If you are currently upgrading CICS Transaction Gateway, ensure that you are aware of all the changes that have been made for this release, and make sure you have made any necessary configuration changes. For more information, see Chapter 4, “Upgrading,” on page 27.

What messages were produced about the problem?

CICS Transaction Gateway writes information, warning and error messages to the message logs (for more information, see “General information about messages” on page 243). Information messages allow you to check that your system is working correctly; warning and error messages inform you about problems. If warning or error messages were produced when CICS Transaction Gateway started, or while

the system was running, these might indicate the cause of the problem.

What software components have been changed since the last successful run?

If you have installed new versions of software components, or a new or modified application, check for warning and error messages. Consider backing out the changes and see if the problem still occurs.

What administrative changes have been made since the last successful run?

If you have changed your CICS TG configuration or changed any CICS resources check that the changes have not caused any warning or error messages. Also check the configuration of the client application. For more information, see Chapter 5, "Configuring," on page 31.

What service changes have been applied since the last successful run?

If you have applied a fix pack, check that it installed successfully and that you did not receive any warning or error messages during installation. Also consider any service changes that have been applied to other programs, which might affect CICS Transaction Gateway.

Review the documentation that was supplied with the fix pack to ensure that the instructions were followed correctly. If the fix pack was installed correctly, try uninstalling it and see if the problem still occurs.

Is the problem related to a particular client application?

If you can identify a client application that is always in the system when the problem occurs, check it for coding errors. If the client application has not yet run successfully, examine it carefully to see if you can find any errors. If you have made changes to the client application since it last ran successfully, examine the new or modified part of the application. Consider the functions of the client application that might not have been fully exercised before.

Is the problem related to system loading?

If the problem seems to be related to system loading, the system might be running near its maximum capacity, or it might be in need of tuning. Check that you have defined sufficient resources (for example, connection manager threads and worker threads). Typically, if you had not defined sufficient resources, you might find that the problem is related to the number of users of the application.

Does the problem occur at specific times of day?

If the problem occurs at specific times of day, it could be dependent on system loading. Typically, peak system loading is at mid-morning and mid-afternoon, so those are the times when load-dependent problems are most likely to happen. Use the CICS TG interval statistics to determine when peak loading occurs and the resource usage at the time; for more information, see "Statistics" on page 259.

Regular backup jobs or other system maintenance might also cause unexpected problems at specific times of day.

What to do next

If preliminary checks have revealed the cause of the problem, you should now be able to resolve it, possibly with the help of other information in the CICS Transaction Gateway information center. If you have not yet found the cause of the problem, you must start to investigate it in greater detail.

To investigate the problem in more detail, begin by deciding the best category for the problem, for example is the problem related to installation, configuration or performance? Then go to "Dealing with problems" on page 229 where you will see a list of problems organized into the various categories. Each topic covers a single problem, and provides details on the symptom, probable cause and action to take.

If the problem is not listed in the categories, you might need to use one of the "Problem determination tools" or you might need to refer to "Problem solving and support" on page 254.

Problem determination tools

Various tools are available for Java debug, JVM dump, system dump, tracing, testing connections, and viewing the logs. TCP/IP diagnostic commands can also be used during problem determination.

For more information about trace see "Tracing" on page 245.

Java debug tools

Links are provided in the Java installation to online information on Java diagnostic tools.

For information about Java diagnostics tools and other tools:

1. Go to the Java installation directory that contains the *IBM SDK and Runtime Environment for Java User Guide*, for example:

   ```
   C:/Program Files/IBM/Java60/docs/en/sdkandruntimeguide.win32.en.htm
   ```

2. In the table of contents click "Contents of the SDK and Runtime Environment".

For additional information on Java diagnosis see: . `http://www.ibm.com/developerworks/java/jdk/diagnosis/60.html`

JVM dump and system dump

JVM dumps and system dumps provide detailed information about the internal status of an IBM JVM, and the configuration of a running CICS Transaction Gateway.

JVM dumps provide a snapshot of a Java Runtime Environment (JRE). System dumps provide a snapshot of the JRE at a process level and also provide diagnostic information regarding the system status and configuration.

For more information, see "Dumping diagnostic information" on page 199.

On some Java Virtual Machine (JVM)s you can force Java to write a stack dump showing the states of the current threads.

For example, on IBM Java SDK you can send a **SIGQUIT (-3)** signal to a Java process to make it write a stack dump to `stderr`. This shows the states of the current threads. Do not do this on a working production system but only on a system which is completely locked.

VTAM buffer trace

VTAM buffer tracing enables you to record the flow of data between logical units in the CICS environment. The information in the trace entries includes the netname of the terminal (logical unit) to which they relate.

For more information see VTAM problem determination in the Networking on z/OS section of the z/OS Basic Skills Information Center: `http://publib.boulder.ibm.com/infocenter/zos/basics/index.jsp`

APING utility

APING is the APPC equivalent of the TCP/IP PING command and provides a way of testing SNA connections.

APING exchanges data packets with a partner computer over APPC and measures the time taken for data transfer. APING can be used to get a first estimate of the session setup time between two computers, and the throughput and turnaround time on that SNA session.

You can use APING to determine whether a session can be set up between two computers and to display extensive error information if session allocation fails. APING consists of two transaction programs: APING: which runs on the client, and APINGD which runs on the server.

For more information see the section on APING in the IBM Personal Communications Information Center at: `http://publib.boulder.ibm.com/infocenter/pcomhelp/v6r0/index.jsp`

TCP/IP diagnostic commands

Use the TCP/IP diagnostic commands for displaying network configuration details, statistics and other information. These commands can be useful during problem determination.

Command	Purpose
arp	Display or modify IP-to-Ethernet or token ring physical address translation tables used by address resolution protocol (ARP).
hostname	Display workstation host name.
ifconfig	Display all TCP/IP network configuration values. This is useful when determining whether or not an IP interface is active. This command is the Linux equivalent of ipconfig.
ipconfig	Display all TCP/IP network configuration values. This is useful when determining whether an IP interface is active or not.
netstat	Display protocol statistics and TCP/IP network connections. This is used for obtaining information about your own IP interfaces, for example, listing IP addresses and TCP/IP routing tables used on your workstation.
nslookup	Display information on Domain Name System (DNS) name servers.
ping	Verify connection to a remote computer or computers. The equivalent command for IPv6 is ping6.

Command	Purpose
traceroute	Trace TCP/IP path to a requested destination. This is useful for determining whether a problem exists with an intermediate node or not.

Dealing with problems

The problems in this section are organized into categories, for example installation, configuration, and performance. Each topic covers a single problem and provides details of the symptom, probable cause, and the action to take.

Installation problems

Problems installing CICS Transaction Gateway.

Installation fails when using AIX WPARs

Installation of CICS Transaction Gateway in the Global Environment can fail if product components are already running within a detached system WPAR (Workload Partition).

Symptom

The installer exits without installing CICS Transaction Gateway, and the following message is written to the installation log:

```
ERROR - A version of IBM CICS Transaction Gateway on your system is currently
running.
```

For more information see “Location of the installation logs” on page 24.

Probable cause

When trying to install the CICS Transaction Gateway into the Global Environment, the installer detected that a version of CICS Transaction Gateway is already running on your system in a detached system WPAR.

Action

Stop the version of CICS Transaction Gateway that is already running in the detached system WPAR and rerun the installer in the Global Environment.

Installation fails if a component is already running

Installation can fail if product components are already running.

Symptom

The installer exits without installing the product, and the following message is written to the installation log:

```
ERROR - A version of IBM CICS Transaction Gateway on your system is
currently running.
```

For more information see “Location of the installation logs” on page 24.

Probable cause

The installer has detected that a version of CICS Transaction Gateway is already running on your system.

Action

Stop the version of CICS Transaction Gateway that is already running and run the installer again.

Startup and shutdown problems

Problems when starting and stopping CICS Transaction Gateway.

Gateway startup problem on 64-bit Linux

Symptom

When attempting to start the Gateway daemon on a 64-bit Linux operating system, startup might fail with the following error:

```
[root@myhost]# ctgstart
cicscli: error while loading shared libraries: libncurses.so.5:
cannot open shared object file: No such file or directory
cicscli: error while loading shared libraries: libncurses.so.5:
cannot open shared object file: No such file or directory
ctgstart - CICS Transaction Gateway start program
(C) Copyright IBM Corporation 1996, 2011.  All rights reserved.
06/14/11 10:20:44:399 [0] CTG6400I CICS Transaction Gateway
is starting
06/14/11 10:20:44:399 [1] CTG6765E The Gateway daemon is unable
to load the CICS TG JNI native library DLL libctgjni.so; the
reason for the load failure is : 'ctgjni (libncurses.so.5:
cannot open shared object file: No such file or directory)'
06/14/11 10:20:44:402 [0] CTG6511I Gateway daemon has shut down
cicscli: error while loading shared libraries: libncurses.so.5:
cannot open shared object file: No such file or directory
```

Probable cause

The 32-bit ncurses-lib package is not installed on the machine.

Action

Install the 32-bit ncurses-lib Linux package for your distribution; the Gateway daemon should now start successfully.

Linux compiler problem

Symptom

When compiling the C and C++ samples on a 64-bit Linux operating system, the samples might fail with one of the following errors:

```
[root@myhost]# make -f samp.mak
Creating Basic ECI C Sample
cc -c -g -DCICS_LNX -m32 -I../../../include -I../../include
-I..  ecib1.c
In file included from /usr/include/features.h:385,
                 from /usr/include/stdio.h:28,
                 from ecib1.c:39:
/usr/include/gnu/stubs.h:7:27: error: gnu/stubs-32.h: No such
```

```
file or directory
make[1]: *** [ecib1] Error 1
make[1]: Leaving directory `/opt/ibm/cicstg/samples/c/eci'
[root@myhost]# make -f samp.mak
Creating Basic ECI C++ Sample
c++ -c -g -DCICS_LNX -m32 -I../../include -I../../../include
ecib1.cpp
In file included from /usr/include/features.h:385,
                 from /usr/include/string.h:27,
                 from ecib1.cpp:39:
/usr/include/gnu/stubs.h:7:27: error: gnu/stubs-32.h: No such
file or directory
make[1]: *** [ecib1] Error 1
make[1]: Leaving directory `/opt/ibm/cicstg/samples/cpp/eci'
[root@myhost]# make -f samp.mak
Creating Basic ECI C++ Sample
c++ -c -g -DCICS_LNX -m32 -I../../include -I../../../include
ecib1.cpp
c++  -o ecib1 ecib1.o -L../../../lib -m32 -lpthread -lc -lcclcp
/usr/bin/ld: skipping incompatible
/usr/lib/gcc/x86_64-redhat-linux/4.4.5/libstdc++.so
when searching for -lstdc++
/usr/bin/ld: skipping incompatible
/usr/lib/gcc/x86_64-redhat-linux/4.4.5/libstdc++.a
when searching for -lstdc++
/usr/bin/ld: skipping incompatible
/usr/lib/gcc/x86_64-redhat-linux/4.4.5/libstdc++.so
when searching for -lstdc++
/usr/bin/ld: skipping incompatible
/usr/lib/gcc/x86_64-redhat-linux/4.4.5/libstdc++.a
when searching for -lstdc++
/usr/bin/ld: cannot find -lstdc++
collect2: ld returned 1 exit status
make[1]: *** [ecib1] Error 1
make[1]: Leaving directory `/opt/ibm/cicstg/samples/cpp/eci'
```

Probable cause

The 32-bit glibc-devel package is not installed on the machine.

Action

Install the 32-bit glibc-devel and libstdc++ Linux packages for your distribution; the applications should now compile successfully.

Gateway daemon fails to shut down

During the initiation phase of a normal shutdown, some calls and requests prevent the shutdown from completing.

Symptom

The Gateway daemon fails to shut down normally (quiesce) or fails to shut down in the expected time.

Probable cause

Some outstanding requests that are waiting to complete are preventing normal shutdown.

Action

- If there are any active applications or tasks in "wait" state in CICS, you must investigate these. For example, to query a CICS task that is in "wait" state, use the CEMT INQ TASK command. For more information about tasks that are in "wait" state see the CICS Transaction Server Information Center at: `https://publib.boulder.ibm.com/infocenter/cicsts/v4r1/index.jsp`
- If normal shutdown fails you can promote this to an immediate shutdown.

Normal shutdown is blocked by an API call

Some API calls that are waiting to finish prevent a normal shutdown of CICS Transaction Gateway.

Symptom

The Gateway daemon fails to shut down normally (quiesce).

Probable cause

API calls that have not yet completed are preventing normal shutdown.

All other ECI or EPI calls that are waiting to finish will prevent the Gateway daemon from quiescing.

Action

Wait for the API calls to complete or perform an immediate shutdown.

The following API calls do not block a normal shutdown of CICS Transaction Gateway:

ECI_GET_REPLY_WAIT

ECI_GET_SPECIFIC_REPLY_WAIT

EPI_GET_EVENT and waitState is EPI_WAIT (an EPIRequest.getEvent call that has its second parameter set to EPI_WAIT causes the request object to wait for events)

Problems starting clients and terminals

cicsterm fails to connect to the CICS server:

The Client daemon can connect to the server, but cicsterm is unable to connect.

Symptom

The **`cicscli -s=servername`** command connects successfully, but **`cicsterm -s=servername`** command does not connect.

Probable cause

The CTIN transaction might not be defined or sign-on capable terminals are not supported by the CICS server. Another possibility is that the CICS server does not have 3270 terminal support over TCP/IP.

Action

1. Check that the CTIN transaction is defined on the server.

2. Issue the **cicsterm -s=servername -a** command to install a terminal that is not sign-on capable. If this is successful, the server probably does not support sign-on capable terminals. cicsterm attempts to install a sign-on capable terminal by default.
3. Check that EPI is supported over the server connection being used.

For more information see "cicsterm options" on page 214.

Configuration problems

Problems with the way that CICS Transaction Gateway is configured.

Problems running the Configuration Tool

Symptom

The Configuration Tool does not display all characters correctly and the operating system has issued warning messages about fonts.

Probable cause

Fonts needed for the code page for the current locale have probably not been installed.

Action

Check the console where CICS Transaction Gateway was invoked.

Refer to the documentation supplied with your operating system and JRE for information on which font packages are required for which locales.

Problems with Client applications

An application fails to start.

Symptom

On Linux systems, your application might fail to start with a message similar to this:

```
relocation error: ./ecib1: undefined symbol:
__6CclBufPCcQ26CclBuf12DataAreaType
```

Probable cause

This can occur with C++ applications compiled with GNU C/C++ compiler (gcc) 2.96. This compiler is not supported.

Action

Recompile your application with a supported compiler listed at "Supported software" on page 9.

CICS connection problems

Problems with connections to CICS Transaction Server.

Unable to connect over SNA

There are several possible reasons why CICS Transaction Gateway is not able to connect to CICS over SNA; for example there might be a problem with an SNA server, or there might be a CICS Transaction Gateway configuration problem. Alternatively, there might be a problem with the connection definition in CICS.

Symptom

CICS Transaction Gateway cannot connect to CICS over SNA.

Probable cause

- CICS Transaction Gateway has not been configured for SNA correctly, or there is a problem with the configuration.
- There is a communications problem between the client and the communications server, when using a remote SNA client.
- There is a problem with the SNA communications server.

Action

- Check the Client daemon log for errors that indicate there is a configuration problem. Also check the SERVER section of the CICS Transaction Gateway configuration file. For more information see "SERVER section of the configuration file" on page 119.
- If you are using the Remote API Client (for SNA), check the local "SNA error log" on page 245.
- If you are using an SNA communications server, check the SNA communications server SNA error log.

For information on configuring the Client daemon to use SNA connections see "Configuring an SNA CICS Server definition" on page 66 and "Defining SNA connections on CICS Transaction Server for z/OS" on page 64.

CCIN or CTIN transactions not recognized

CCIN or CTIN transactions have not been recognized.

Symptom

If CCIN is not installed correctly on the CICS server the following symptoms occur:

- TCP/IP and SNA connections can not be established to that server.
- Client terminal installs fail on that server.

Probable cause

CCIN and CTIN have not been installed on the server. The CCIN transaction installs a Client connection on the CICS server when using either the TCP/IP or SNA protocol. The CTIN transaction installs your client terminal definition on the CICS server.

Action

If you are using TCP/IP or SNA, the CICS server must have CCIN installed. CTIN must also be installed if you require CICS 3270 emulation and are running in a supported configuration.

IPIC connection problems

A problem can occur if a capability exchange is not valid, if a handshake failure occurs, or if the system cannot acquire an IPCONN.

Attempting connection to CICS on wrong TCP/IP port:

If CICS Transaction Gateway attempts to connect to CICS on the wrong TCP/IP port an error occurs.

Symptom

The following error is returned:

```
ECI_ERR_NO_CICS
```

Probable cause

CICS Transaction Server is listening on a different TCP/IP communications port to the one through which CICS Transaction Gateway is attempting the connection. This is because the SERVER section of the CICS Transaction Gateway configuration file (ctg.ini) is specifying the wrong port number.

Action

1. Check which port CICS Transaction Server is listening on.

   On TSO option 6, issue the command:

   ```
   NETSTAT ALLCON (APPLD *CISS*
   ```

   On USS, issue the command:

   ```
   netstat -a -G *CISS*
   ```

   Sample output:

   ```
   IY2GTGA2 0005AD5F Listen
    Local Socket:   1.23.456.789..1120
    Foreign Socket: 2.34.567.890..43066
    Application Data:  DFHIIY2GTGA2CISSIPIC    IP50889
   IY2GTGA2 0005DB97 Establsh
    Local Socket:   1.23.456.789..1120
    Foreign Socket: 2.34.567.890..43066
    Application Data:  DFHIIY2GTGA2CISSIPIC    0000000700000007
   ```

   This example shows that the IPIC TCPIPService is listening on port 50889 and also that an IPCONN is in use. The generated IPCONN name is 00000007.
2. Change the port number in the configuration file (ctg.ini). For more information, see “Port” on page 77.

Additional information

The `Application Data` string in the example contains these values:

DFH The CICS Transaction Server prefix.

I Inbound.

IY2GTGA2
The CICS APPLID.

CISS The listening transaction CISS for inbound IPIC requests.

IPIC The TCPIPService.

IP50889
The TCPIPService name.

0000007
The generated IPCONN name.

IPIC over SSL incorrectly configured:

A problem can occur if SSL has been configured for a connection that does not use SSL.

Symptom

Whilst attempting to establish an IPIC over SSL connection between CICS Transaction Gateway and CICS Transaction Server, the following message appears in the CICS Transaction Server log:

Probable cause

IPIC over SSL is only supported in local mode. Either SSL on CICS Transaction Gateway has been incorrectly configured, or the CICS Transaction Server TCPIPSERVICE definition has been incorrectly configured for SSL.

Action

Ensure you are running IPIC over SSL in local mode and that your configuration is correct. For more information see Chapter 6, “Scenarios,” on page 125.

IPIC connection to CICS fails

The client application receives an ECI_ERR_NO_CICS error when attempting to send a request to CICS over an IPIC connection.

Symptom

An ECI_ERR_NO_CICS error occurs and the following message is written to the CICS Transaction Gateway log:

```
CTG8431E Handshake failure for IPIC connection to CICS server CICSIPIC
response code=ISCER_EXCEPTION, reason=AUTOINSTALL_FAILED [1]
```

The following message is written to the CICS Transaction Server log:

Probable cause

The TCPIPService is configured to use predefined IPCONNs exclusively but a matching IPCONN definition was not found.

Action

Check the IPCONN definitions installed on CICS; look to see if one exists that has an APPLID that matches the APPLID and APPLID qualifier of the Gateway daemon. For more information see “IPIC server connections” on page 46.

Alternatively you can enable autoinstall on the TCPIPService. For more information see `https://publib.boulder.ibm.com/infocenter/cicsts/v4r1/index.jsp`

TCP/IP connection problems

Problems when connecting to CICS over TCP/IP.

A cicsterm command fails:

A cicsterm command fails because 3270 terminal emulation is not supported.

Symptom

The following message is returned to the terminal emulator:

```
CCL7053E Errors found while communicating with server
```

Probable cause

The cicsterm and cicsprnt commands use CICS 3270 emulation. However not all mainframe CICS servers support CICS 3270 emulation over TCP/IP.

Action

Check to see whether your CICS server supports terminal emulation. For more information see Chapter 11, “3270 terminal emulation and printing,” on page 213.

Security problems

Problems with security.

SSL problems

SSL problems might include access denied exceptions accompanied by message CTG6651E, and non-recognition of key ring names.

SSL exceptions can occur if a non-valid digital certificate is used, or if the wrong security credentials are sent on a request. If the Gateway daemon uses a non-valid SSL certificate, a DFHIS2040 message is generated when the connection is established:

```
CTG6651E Unable to connect to the Gateway daemon: [address = IP address ,
port = port ] [error ]"
```

If an SSL exception occurs, enable stack tracing in the CICS Transaction Gateway. Stack tracing indicates what was happening when the exception occurred. It also provides information about the configuration, such as the value of the CLASSPATH environment variable. If this does not give you enough information to diagnose the problem, obtain a standard trace and contact your IBM support organization.

For more information see “Exception stack tracing” on page 111.

Application receives an “access denied” exception:

An application is not able to read from the file system containing the keystore.

Symptom

An application receives a message similar to this:

```
java.io.IOException: CTG6651E: Unable to connect to the Gateway.
                     [address = killerb2b, port = 8050]
                     [java.security.AccessControlException: access denied
                     (java.io.FilePermission \jssekeys\testclient.jks read)]
```

Probable cause

The application is running with Java security enabled and does not have permission to read from the file system containing the keystore.

Action

Add a FilePermission for the location of the keyring file.

SSL handshake failure:

An SSL handshake failure can occur if an IPCONN is not configured to use SSL in some situations.

Symptom

This problem results in an ECI_ERR_NO_CICS error.

Probable cause

The IPCONN definition is not configured to use SSL.

Action

Configure your IPCONN definition to use SSL.

Identity propagation not supported

A security exception and message CTG9631E occurred when a back-level CICS server that does not support identity propagation was being used.

Symptom

The following message is returned as an API return code or as an exception to the EJB:

Probable cause

Work is being passed to a back-level CICS server, which does not support identity propagation, resulting in an ECI_ERR_SECURITY_ERROR return code.

Action

Use a level of CICS that supports identity propagation. For more information, see "Support for identity propagation" in the *CICS Transaction Server V4.1 Information Center*.

Security violation during identity propagation

A security violation and message DFHIS1027 occurred during identity propagation.

Symptom

The following message appears in the CICS Transaction Server log:

Probable cause

The IPIC connection is incorrectly set to use VERIFY user authentication.

Action

Modify the IPCONN definition for the IPIC connection referred to in message DFHIS1027; change the user authentication setting from USERAUTH=VERIFY to USERAUTH=IDENTIFY.

RACF mapping problem during identity propagation

A RACF mapping problem and message ICH408I occurred during identity propagation.

Symptom

The following message appears in the z/OS system log:

```
ICH408I USER userid GROUP group NAME userid owner DISTRIBUTED IDENTITY IS
NOT DEFINED: distinguished_name realm_name
```

Probable cause

RACF does not contain a mapping that associates the distinguished name of the user with a RACF user ID.

Action

If the user is permitted to access the CICS resources, create a RACF mapping that includes the distinguished name of this user. For more information see "Configuring RACF for identity propagation" in the CICS Transaction Server Information Center at: `https://publib.boulder.ibm.com/infocenter/cicsts/v4r1/index.jsp`

Identity propagation login module not enabled

The CICS Transaction Gateway identity propagation login module is not enabled and verification fails with an IRR012I message.

Symptom

The following message appears in the z/OS system log:

```
IRR012I VERIFICATION FAILED. USER PROFILE NOT FOUND
```

Probable cause

The CICS Transaction Gateway identity propagation login module is not enabled.

Action

Enable the CICS Transaction Gateway identity propagation login module in WebSphere Application Server.

Memory problems

Problems cause by insufficient memory being available.

Memory use increases over time

The amount of memory used by the Gateway daemon might increases over time and a java.lang.OutOfMemory exception might occur.

The maximum number of connection manager threads and worker threads is defined in the CICS Transaction Gateway configuration.

Symptom

The Gateway daemon stops responding and the JVM writes a java.lang.OutOfMemory exception to the stderror log file or to the Java dump file. The JVM also creates various dump files in the information log. There is probably no noticeable decrease in performance before the problem occurs. If you happened to be monitoring memory usage before the dump occurred, you would have seen that memory usage gradually increased over time until eventually the limit was reached.

Probable cause

- There is a problem with a user-written application, for example a request exit which has remained inadvertently connected and is using Java resources.
- There are too many active Java threads (connection manager threads and worker threads).
- The Java heap size is unnecessarily large. Because the memory required to create Java heap and Java threads is allocated from the same finite storage area, it is possible that making the Java heap too large could indirectly cause a java.lang.OutOfMemory exception because there would then be insufficient memory available to create enough Java threads.
- The Java heap size is too small.

Action

- If there is a problem with a user application, ensure that the application practices good memory management techniques, such as freeing resources when they are no longer required. To limit the size of the trace output file, use the **ctgstart -tfilesize** option. You can also use the **ctgstart -truncationsize** option to reduce to zero the size of data blocks in the trace. This reduces the effect on performance.
- If the Java heap size is unnecessarily large or too small, set the maximum amount of heap memory available to the JVM by using the -Xmx option. The default heap size specified by the CICS Transaction Gateway is 128MB.
- Run a memory usage monitor against the Gateway daemon process.
- Set the amount of memory that Java allocates to each thread by using the -Xmso and -Xss options. The maximum size of the Java stack and the native stack is the sum of these two values. When using the IBM Runtime Environment, Java 2 Technology Edition, Version 6, the default values for the -Xmso and -Xss options are 256KB, on most platforms. Do not change the Java stack and native stack sizes from their default values.

Additional information

The way that Java allocates memory depends on your JVM implementation. Most JVMs allow you to adjust the maximum amount of heap memory and adjust the amount of memory allocated to each thread.

For more information on thread limits see “Threading model” on page 176. For more information on Java memory allocation and JVM stack sizes, see the *IBM Java Diagnostics Guides Information Center*.

Also see “Avoiding out of memory conditions” on page 187.

Related reference:

“List of statistics” on page 269
These statistics are available from the CICS Transaction Gateway.

Related information:

“Displaying statistics” on page 265
You can use the **ctgadmin** command to display statistical information about the CICS Transaction Gateway, or obtain statistics using either the C or Java Statistics API interface.

“Gateway daemon resources” on page 70
Use the CICS Transaction Gateway configuration tool to configure the Gateway daemon resources, or edit the GATEWAY section of the configuration file directly.

Performance problems

Problems with system performance.

Client daemon stops when CICS task limit is reached

A CPMI transaction or an equivalent mirror transaction task has locked in a CICS server and cannot send data back to CICS Transaction Gateway.

Symptom

The Client daemon appears to stop responding.

Probable cause

The MAXTASKS limit of the CICS server might have been reached. This problem prevents the mirror program from returning data to the Client daemon, which appears to stop responding. The problem typically occurs when CICS has a high number of concurrent transactions.

Action

Put the mirror transaction in a TranClass that has a MAXACTIVE value that is less than the MAXTASKS value of your CICS server. All new requests to the CICS server are queued, and CICS can continue to process current requests. The value that you should specify for MAXACTIVE depends on your installation, and other tasks running in the server.

Mirror transaction does not time out

A task has suspended in CICS awaiting a response from an ECI client application during an extended LUW.

Symptom

Mirror tasks are left running in CICS.

Probable cause

The default ECI mirror transaction (CPMI or CSMI) uses a PROFILE of DFHCICSA. The profile has an RTIMOUT value of NO. This means that if a request using an extended LUW is suspended in the client application whilst not in an ECI call, the CICS mirror transaction fails to time out.

Action

To enable the mirror transaction set RTIMOUT, in the mirror transaction's profile, to purge extended LUW tasks hanging in the client application. This should ensure that if no response is received from the client application after the timeout period has elapsed CICS purges the mirror transaction and rolls back the associated unit of work. It is the RTIMOUT value that causes CICS to purge the mirror.

Corrupted data when using channels and containers

Data corruption when using channels and containers can occur if an incorrect CCSID is specified.

Symptom

Unexpected or corrupt data is returned to the client application when using an IPIC connection and channels and containers.

Probable cause

- The wrong CCSID is specified on the client application channel and has been inherited by the container.
- The wrong CCSID is specified on the container.

Action

1. If corrupted or unexpected data is returned, run a Gateway daemon trace to find out which code page the JVM is running on. Look in the System Properties section at the top of the trace.
2. For Java applications, use the setCCSID method to set the required code page on the channel. You must explicitly specify a CCSID when creating the container. For C or .NET applications, specify a CCSID when creating a CHAR container.

For more information on how to find the code page that the Client has sent to the server, see “Data conversion,” on page 285.

Resource problems

Problems due to shortage of resources.

Shortage of IPIC resources on the CICS server

An error can occur if there is a shortage of IPIC resources.

Symptom

Intermittent ECI_ERR_RESOURCE_SHORTAGE errors occur when sending an ECI request to CICS over IPIC.

Probable cause

All the defined sessions for the connection are in use. Each active session uses one CICS task, so the maximum number of sessions allowed is 999. CICS Transaction Gateway allocates 300 KB of memory for each session. If all the defined sessions are in use, any new requests receive an ECI_ERR_RESOURCE_SHORTAGE error.

Action

- Increase the SENDSESSIONS value in the CICS Transaction Gateway configuration file (ctg.ini).
- Increase the IPCONN ReceiveCount value in CICS.

For more information see "Configuring IPIC on CICS Transaction Server for z/OS" on page 49

Java problems

Problems related to Java.

JVM dumps and system dumps provide detailed information about the internal status of an IBM JVM, and the configuration of a running CICS Transaction Gateway.

JVM dumps provide a snapshot of a Java Runtime Environment (JRE). System dumps provide a snapshot of the Java Runtime Environment at a process level and also provide diagnostic information regarding the system status or configuration.

For more information, see "Dumping diagnostic information" on page 199.

"Access denied" security exception

If an application program attempts a task that is protected by the Java 2 Security Manager the result is an "access denied" security exception.

Symptom

When an application is using the ECI or EPI interface in a Java 2 Security Manager environment, the following exception occurs:

```
java.security.AccessControlException: access denied
(java.util.PropertyPermission * read,write)
```

Probable cause

The application program is attempting a task that is protected by the Security Manager.

Action

Refer to the information on security permissions required by programs running in this environment. For more information see *CICS Transaction Gateway for Multiplatforms: Programming Guide.*

General information about messages

Information about message locations, formats, and prefixes.

Message locations

The Gateway daemon and Client daemon use different logs, the configuration file defines where log messages are written. See "Gateway daemon logging" on page 74 and "Client daemon logging" on page 87 for more information.

Message format

Messages have the following format:

```
CTGnnnnt: <message text>
```

where `nnnn` is a number, and `t` is one of the following:

Identifier	Purpose of message	Written to
I	information	Information logs
E	error	Error logs
W	warning	Error logs

Message redirection

All CICS Transaction Gateway messages can be optionally redirected to standard error, and standard error can be written to a file called `outputfile`. To do this, use the following command:

```
ctgstart 2>outputfile >&2
```

For more information about redirecting messages, see the documentation for your operating system.

Message prefixes

CICS Transaction Gateway messages have the prefix CTG. Client daemon messages have the prefix CCL.

For an explanation of all CICS Transaction Gateway messages, see the *CICS Transaction Gateway: Messages* book.

API errors

Error codes resulting from incorrect use of the APIs are returned to the associated applications. Applications must notify the user about such errors, and must provide information on the required user response.

Telnet clients

Telnet clients can cause a problems with the display of information, for example by truncating lines of text in messages.

If you are using Telnet, sometimes message text lines that exceed a certain length are truncated.

If you run the CICS Transaction Gateway **cicsterm** command from a Telnet prompt, certain Telnet clients can cause problems with the display such as truncation. This is usually a problem with the Telnet client that you are using, or the terminal type that you are emulating. Currently there is no solution to this problem.

SNA error log

The SNA error log can help your support organization diagnose problems.

The default installation log locations are:

Platform	Server log location	Client log location
AIX	/var/sna/sna.err	/var/sna/sna.err
HP-UX	/var/opt/sna/sna.err	/var/opt/sna/sna.err
Linux	/var/opt/ibm/sna/sna.err	/var/opt/ibm/sna/sna.err
Solaris	/var/opt/sna/sna.err	/var/opt/sna/sna.err

If you are using an IBM Remote API Client for Windows to communicate with an IBM Communications Server on Unix or Linux, the default location for the SNA Client log is C:\IBMCS\w32cli\sna.err

The user who starts the Client daemon must have the authority to write to the SNA error log. The IBM Communications Server for Linux uses /var/log/warn if the user does not have authority to write to the SNA error log.

Tracing

Tracing can be enabled and controlled for different components of the CICS Transaction Gateway.

Tracing, especially debug tracing, decreases performance.

Gateway daemon tracing

Gateway daemon tracing can be set in the Gateway daemon configuration file, or with a command option.

For information about controlling trace at run time using the ctgadmin command see "Trace options" on page 198.

To enable standard trace when starting the Gateway daemon use this command:

```
ctgstart -trace
```

To enable debug trace, use this command:

```
ctgstart -x
```

For more information on starting the Gateway daemon with trace options, see "Starting and stopping the Gateway daemon" on page 193.

Specifying trace output destination

You can use the Configuration Tool to define a default destination for trace output. See Using the Configuration Tool for more information.

If you do not define a default destination for trace output, trace output is written to `stderr` by default.

To override the default destination for trace output, use the **ctgstart -tfile** option.

To start the Gateway daemon with debug tracing enabled and write the trace output to the file specified in the *filename* variable, use this command:

```
ctgstart -x -tfile=filename
```

Gateway daemon trace levels

There are three main levels of Gateway daemon tracing: stack trace, standard trace, and debug trace.

Stack tracing
Trace entries are written only when a Java exception occurs. They can help to determine the source of the exception. Use this when it is important to maintain performance. See the following sample stack trace.

Standard tracing
Java exceptions and the main Gateway daemon functions and events are traced. By default, the Gateway daemon displays only the first 128 bytes of any data blocks (for example the COMMAREA, or network flows) in the trace.

Debug tracing
Java exceptions and the main Gateway daemon functions and events are traced in greater detail than with stack or standard tracing. By default, the Gateway daemon fully outputs any data blocks in the trace. Use this only when performance is not important or if standard tracing did not give enough information to solve the problem.

Client daemon tracing

Client daemon tracing is a useful problem determination tool for resolving communication problems.

You can use the trace functions to collect detailed information on the execution of a particular function or transaction. A trace can show how the execution of a particular activity is affected by, for example, the execution of other tasks in a CICS system. Each trace entry has a time stamp, which provides information on the time taken to perform certain activities.

To learn how to turn tracing on, see “Starting client tracing” on page 206.

For information on specifying the components of the Client daemon to be traced, see the cicscli -m command.

The output from the trace function is a binary trace file called, by default, cicscli.bin in the /var/cicscli subdirectory. You can specify a different name for this file, using the Configuration Tool. However, you cannot change the .BIN extension. Using the **Client trace file wrap size (KB)** configuration setting, you can specify that the binary trace file should wrap into a second trace file, and you can also specify the maximum size of these files.

To read the trace, run the cicsftrc utility to convert the binary file or files into a text file. This text file is called cicscli.trc by default. The default trace files are:

cicscli.bin
The binary trace file produced by running the Client daemon trace.

cicscli.wrp
The second binary trace file if wrapping of client trace is enabled.

cicscli.trc
The name of the text trace file produced when the binary trace file is converted to a text file using the cicsftrc utility.

cicscli.bak
The backup file of the binary trace file. A backup file is produced from any existing .BIN file when you turn tracing on.

cicscli.bbk
The backup of the first binary trace file if memory mapped tracing is enabled.

cicscli.wbk*n*
The backup of subsequent binary trace files, if memory mapped tracing is enabled, where *n* is the number of the original .WRP file.

See "Formatting the binary trace file" on page 248 for information on the trace conversion utility.

Wrapping the Client trace

You can control the size of the binary trace file by specifying that it wraps into a second trace file. Use the **Client trace file wrap size (KB)** configuration setting to turn on wrapping trace; specify the maximum size of the wrapping trace (in kilobytes). If this value is 0 (the default), wrapping trace is not turned on.

When wrapping trace is turned on with standard I/O tracing, two files (called cicscli.bin and cicscli.wrp) are used. Each file can be up to half the size of the **Client trace file wrap size (KB)** value.

Memory mapped tracing

If you enable wrapping trace, you can use the `-b` switch when you issue the `cicscli` command to turn tracing on. This specifies that memory mapped trace files should be used.

With memory mapped tracing, the operating system's paging mechanism is used to swap data between memory and the trace file. This improves performance significantly when compared to standard file I/O, because the trace file is opened and written to less frequently. Because the operating system is responsible for flushing data to disk, data is not normally lost if an application terminates unexpectedly. However, if the operating system itself fails, data can be lost, and the trace file can be corrupted. If you are diagnosing problems where the server fails and needs to be restarted, use standard I/O tracing instead of memory mapped tracing.

If you use memory mapped tracing, the size of the trace files is limited to 10 MB, and in addition to cicscli.bin and cicscli.wrp, you might see a series of files of the form cicscli.wrp1, cicscli.wrp2...cicscli.wrp*n*, where *n* is the number of files needed to hold the total amount of trace data specified in the Client trace file wrap size (KB) field of the Configuration Tool; see "Client trace file wrap size (KB)" on page 111 for the maximum amount of data that can be specified. The trace formatter finds all files in the sequence when you format the binary files. Memory mapped tracing uses up to 10MB of memory.

See "Starting client tracing" on page 206 for details of how to issue the command, and "Security considerations for trace and log files" on page 208 for important information on security.

Formatting the binary trace file

If you are using memory mapped tracing, note that data is not always written to disk immediately. As a consequence, turn tracing off before you format the binary trace file, to ensure that all data is flushed to disk.

You use the Binary Trace Formatter utility cicsftrc to convert the binary trace file cicscli.bin to ASCII text. The utility has the following parameters:

-m=*list of components*
: Specifies that only trace points from the listed components are written to the text file. The components you can specify are the same as for cicscli -m; see the cicscli -m command. If -m is not specified, all trace points in the binary trace are written to the text file.

-w[=*filename***]**
: Indicates that there are two or more binary trace files to format and then concatenate (that is, the binary files were created with a wrapping trace). If no file name is specified with the -w parameter, cicsftrc assumes that the name of the second trace file is cicscli.wrp.

-n Indents entry and exit points in the test trace file to make it more readable. By default, indentation is turned off.

-d Specifies detailed trace formatting. If you are using EPI calls, cicsterm or CICSPRINT, an approximation of the screen layout will be included in the trace.

-i=*filename*
: Specifies the name of the input (binary) trace file, which is cicscli.bin by default.

-o=*filename*
: Specifies the name of the output (text) trace file. If no -o parameter is specified, the name of the text trace file is assumed to be cicscli.trc.

-f Overwrite any existing files.

-s Do summary trace formatting. Summary trace formatting is controlled by a template file (cclsumtr.txt), which is read in at initialization time. It formats key trace points, and shows for example the flow of user API calls, the progress of calls through the Client daemon, and network flows to the server. For the most detailed results, specify the API.2 component when you define the components to be traced. Summary tracing provides an overview; use it as requested by your IBM support organization.

Note: Versions of the CICS Transaction Gateway earlier than 5.0.1 cannot format binary trace files produced by this version.

```
       -->Sample of API summary trace taken with API.2 and DRV options.

[Process ,Thread  ]     Time       API Summary                   CCLCLNT Summary               Comms Summary
=============================================================================================================
...
...
...
[000000bf,0000017c] 12:08:32.190 --->[7315] CCL3310 ECI Call type ECI_SYNC, UOW=0
[00000089,000000a4] 12:08:32.290                                                -S->[4410] CCL4411 TCP/IP (to STEMPLAR) send data: Length=89
[00000089,00000063] 12:08:32.400                                                <-R-[4418] CCL4412 TCP/IP (to STEMPLAR) receive data: Length=12
[00000089,0000018b] 12:08:32.511                                                <-R-[4418] CCL4412 TCP/IP (to STEMPLAR) receive data: Length=29
[00000089,000000a4] 12:08:32.521                                                -S->[4410] CCL4411 TCP/IP (to STEMPLAR) send data: Length=94
[00000089,000000a4] 12:08:32.531                                                -S->[4410] CCL4411 TCP/IP (to STEMPLAR) send data: Length=94
[00000089,000000a4] 12:08:32.541                                                -S->[4410] CCL4411 TCP/IP (to STEMPLAR) send data: Length=94
[00000089,000000a4] 12:08:32.541                                                -S->[4410] CCL4411 TCP/IP (to STEMPLAR) send data: Length=94
[00000089,000000a4] 12:08:32.551                                                -S->[4410] CCL4411 TCP/IP (to STEMPLAR) send data: Length=94
[00000089,00000168] 12:08:32.581                                                <-R-[4418] CCL4412 TCP/IP (to STEMPLAR) receive data: Length=12
[00000089,0000017e] 12:08:32.601                                                <-R-[4418] CCL4412 TCP/IP (to STEMPLAR) receive data: Length=31
[000000bf,0000018e] 12:08:32.621     [7364] CCL3350   Event Service Thread got a request REQUEST_TYPE_ECI_1
[000000bf,0000008a] 12:08:32.671 <---[7316] CCL3311 ECI Call type ECI_SYNC, UOW=0 got rc=ECI_NO_ERROR {Time in API = 0.821 seconds}
```

Figure 20. Sample of API summary trace taken with API.2 and DRV options

Points to note:

1. {Time in API} shows the amount of time that the client API call took to complete. This can help when investigating performance problems; see Chapter 8, “Performance,” on page 173 for more information.
2. The API Summary column refers to client API code inside the user application process. It tracks when user requests enter and leave the client API code. `--->` and `<---` show the program entering and leaving the Client daemon API.
3. CCLCLNT is the background Client daemon process. You get entries here only if you specify the CCL component.
4. The Comms Summary tracks when Client daemon calls enter and leave the network. `-S->` shows a request being sent to the network; `<-R-` shows a reply being received.

If a user application is making EPI calls, or using cicsterm or cicsprnt, the trace formatter puts an approximation of the screen into the trace. The following screen capture is from a formatted trace file, taken from the CECI transaction. It is an aid to problem determination, not a completely accurate representation of the screen. See “Formatting the binary trace file” on page 248 for details of how to format the trace file.

```
Command = Erase/Write, so clearing main screen
 Command2 = Read Modified
 WCC = 0x32 ( Free Kbd,80 char)
 Set Buffer Address to (1,2)
 Insert Cursor @ (1,2)
 Set Buffer Address to (1,1)
 Start Field Extended (Unprotected,Alphanumeric,Display,not-pen-detectable,Foreground Colour Green)
 Data : ' '
 Insert Cursor @ (1,3)
 Set Buffer Address to (2,1)
  Data : 'User        '
  .....
  .....
  .....
  .....
 Set Buffer Address to (24,49)
 Start Field Extended (Autoskip (Prot+Num),Display,not-pen-detectable,Foreground Colour Turquoise)
 Data : '9'
 Set Buffer Address to (24,51)
 Start Field Extended (Unprotected,Alphanumeric,Intense,pen-detectable,Foreground Colour Red)
 Data : 'Messages    '
          1         2         3         4         5         6         7         8
    12345678901234567890123456789012345678901234567890123456789012345678901234567890
 >+--------------------------------------------------------------------------------+
01| -                                                                              |
02| ´STATUS. . :´Enter one of the following:                                       |
03|     ´                                                                          |
04| ´ABend       EXtract     READPrev    WAit                                      |
05| ´ADdress     FEpi        READQ       WRITE                                     |
06| ´ALlocate    FOrmattime  RECeive     WRITEQ                                    |
07| ´ASKtime     FREE        RELease     Xctl                                      |
08| ´ASSign      FREEMain    RESetbr                                               |
09| ´Bif         Getmain     RETRieve                                              |
10| ´CAncel      Handle      RETUrn                                                |
11| ´CHange      IGnore      REWrite                                               |
12| ´CONNect     INquire     SENd                                                  |
13| ´CONVerse    ISsue       SET                                                   |
14| ´DELAy       LInk        SIGNOFf                                               |
15| ´DELETE      LOad        SIGNON                                                |
16| ´DELETEQ     PErform     START                                                 |
17| ´DEQ         POP         STARTBr                                               |
18| ´DUmp        POSt        SUspend                                               |
19| ´ENDbr       PUsh        SYncpoint                                             |
20| ´ENQ         READ        Unlock                                                |
21| ´ENTer       READNext    Verify                                                |
22|     ´                                                                          |
23| ´PF´1-Help        ´2-HEX         ´3-End         ´4-EIB         ´5-Variables    |
24|    ´6-User                                      ´9-Messages                    |
  +--------------------------------------------------------------------------------+
  | 1BþC000                  STEMPLAR                                              |
  +--------------------------------------------------------------------------------+
    12345678901234567890123456789012345678901234567890123456789012345678901234567890
             1         2         3         4         5         6         7         8
```

Figure 21. Screen capture from a formatted trace file

The formatter lists the commands that built the screen, and shows an approximation of the screen.

Format of trace entries

The format of the entries in the Client trace file.

time [*process id*,*thread id*] [*number*] *component trace message data*

where:

time
: The time the entry was written, to millisecond accuracy.

[*process id, thread id***]**
: Process ID is a unique number that the operating system uses to identify a process. Thread ID is a unique number that the operating system uses to identify a thread within a particular process.

[*number***]**
A number that uniquely identifies the particular trace entry. This helps your support organization in the diagnosis of serious problems.

[*component***]**
The component of the product to which this entry applies.

trace message
The trace message number and text.

data
Some trace entries include a dump of key data blocks in addition to the trace message.

Sample Client trace

This sample Client daemon trace shows the trace information recorded during the successful connection of a Client daemon to a CICS server using the TCP/IP protocol.

This trace was generated using the commands:

```
cicscli -s=cicstcp -d cicscli -o
```

```
17:03:57.580 [0000080c,00000908] [1007] TRC:CCL1042 *** CICS Client for Windows v7 Service Level 00 - service trace started ***
17:03:57.590 [0000080c,00000908] [2183] CCL:CCL2048 Maximum trace data size set to 512
17:03:57.600 [0000080c,00000908] [2114] CCL:CCL2142 GetNextTimeout timeout is 0 seconds
17:03:58.612 [0000080c,00000908] [2030] CCL:CCL2106 Comms Event : LINK-UP
17:03:58.622 [0000080c,00000908] [2019] DRV:CCL2055 Connection with server established (linkID=1)
17:03:58.622 [0000080c,00000908] [2035] CCL:CCL2109 Send server TCS data
17:03:58.632 [0000080c,00000908] [3214] CCL:CCL3251 Comms Allocate request (LinkId=1, Tran='CCIN')
17:03:58.632 [0000080c,00000908] [3217] CCL:CCL3238 Comms Allocate completed (LinkId=1, ConvId=1, Rc=0)
17:03:58.632 [0000080c,00000908] [2127] CCL:CCL2143 CommsBegin - OK
17:03:58.632 [0000080c,00000908] [3100] CCL:CCL3113 CCIN install request: ApplId='*       ', Code page=819
...
...
...
17:04:00.625 [0000080c,00000908] [3102] CCL:CCL3114 CCIN install response: ApplId='@0Z8AAAA', Code page=8859-1, Rc=0
17:04:00.625 [0000080c,00000908] [3241] CCL:CCL3255 Comms Complete request (ConvId=1)
17:04:00.635 [0000080c,00000908] [3244] CCL:CCL3246 Comms Complete completed (ConvId=1, Rc=0)
17:04:00.635 [0000080c,00000908] [3218] CCL:CCL3252 Comms Deallocate request (ConvId=1)
17:04:00.645 [0000080c,00000908] [3221] CCL:CCL3239 Comms Deallocate completed (ConvId=1, Reason=0, Rc=0)
17:04:00.645 [0000080c,00000908] [2042] CCL:CCL2114 Processed TCS Reply - Server connected
17:04:00.645 [0000080c,00000908] [2114] CCL:CCL2142 GetNextTimeout timeout is 0 seconds
17:04:00.655 [0000080c,00000908] [2114] CCL:CCL2142 GetNextTimeout timeout is 0 seconds
17:04:00.664 [0000080c,00000908] [1004] TRC:CCL1043 *** Service trace ended ***
```

Figure 22. Sample Client daemon trace

Message ID	Explanation
CCL1042	Start of trace message. Each time a trace is started, a backup of the old trace file is created, and the trace file is overwritten. You can delete the file when required. Check the time stamp to ensure that you are reading the correct trace.
CCL2048	Maximum trace data size is at the default size of 512 bytes. You can modify this size by specifying the size value in the start command for the client trace; see the cicscli -d command.
CCL3251	The client sends a CCIN transaction to the server to install its connection definition on the server.
CCL3238	A conversation ID has been successfully allocated. If more than one conversation is active, use the conversation ID to distinguish between them.
CCL3113	The client sends a CCIN transaction to the server with Appl ID set to * to install its application. The Appl ID is specified in the configuration file as `Client=*`. This requests the server to dynamically generate an Appl ID that is unique within the CICS server system.

Message ID	Explanation
CCL3114	This shows the dynamically generated Appl ID.
CCL1043	End of trace message.

The following sample shows trace information recorded when we tried to connect to a CICS server over TCP/IP using an invalid port number. The port number specified in the configuration file file was not defined in the services file of the server. Hence, the connection could not be established.

```
16:16:41.562 [0000093c,000008ec] [1007] TRC:CCL1042 *** CICS Client for Windows v6 Service Level 00 - service trace started ***
16:16:41.572 [0000093c,000008ec] [2183] CCL:CCL2048 Maximum trace data size set to 512
16:16:41.582 [0000093c,000008ec] [2114] CCL:CCL2142 GetNextTimeout timeout is 0 seconds
16:16:41.612 [0000093c,000008ec] [2114] CCL:CCL2142 GetNextTimeout timeout is -1 seconds
16:16:41.622 [0000093c,000008ec] [3207] CCL:CCL3249 Comms Open request (Server=CICSTCP, Driver=CCLIBMIP)
16:16:41.622 [0000093c,000008ec] [4408] DRV:CCL4413 CCL4413 TCP/IP (to CICSTCP) address=192.113.36.200, port=1089, socket=3
16:16:41.622 [0000093c,000008ec] [3210] CCL:CCL3236 Comms Open completed (Server=CICSTCP, LinkId=1, Rc=0)
16:16:41.633 [0000093c,000008ec] [2114] CCL:CCL2142 GetNextTimeout timeout is 3660 seconds
16:16:41.633 [0000093c,000008ec] [1004] TRC:CCL1043 ***Service trace ended ***
```

Figure 23. Client daemon trace: using an invalid port number

Message ID
Explanation

CCL4413
Shows the port number used for this connection request.

You must check your definitions in the SIT on the server, the configuration file on the workstation, and the services file for the port number specified.

You must provide a valid port number or use the default value.

JNI tracing

Enable JNI trace by setting environment variables, or by using a ctgstart override, or when starting an application in local mode.

Use one of the following methods to enable JNI trace:

- When you start the CICS Transaction Gateway, issue the command:
  `ctgstart -j-Dgateway.T.setJNITFile=`*`filename`*

  where *filename* is the name of the file to which trace output is to be sent. If you do not specify a full path to the file, the location is <install_path>/bin.
- Set the following environment variables before you start the CICS Transaction Gateway or Java Client applications running in local mode:

  CTG_JNI_TRACE
  Use this environment variable to set the name of the JNI trace file. This environment variable only defines the name of the JNI trace file; it does not enable trace. JNI trace is output as plain text, and there is no requirement to use a particular extension for the file name.

  CTG_JNI_TRACE_ON
  Set this environment variable to YES (case-insensitive) to enable JNI trace when the CICS Transaction Gateway or Java Client application is started.
- For Java Client applications running in local mode, use Java to launch your application and set the system property gateway.T.setJNITFile, as shown in the following example:
  `java -Dgateway.T.setJNITFile=`*`filename application`*

where
- *filename* is the name of the file to which trace output is to be sent
- *application* is the application to launch

You cannot enable JNI trace through the Configuration Tool.

Tracing Java Client applications

You can enable tracing in the application by using a Java directive when you start the JVM, or by adding calls to the CICS Transaction Gateway tracing API.

Use the **-D** option on the **java** command to specify Java directives. The tracing API comprises several static methods in the T class of the CICS Transaction Gateway. See the information about tracing in Java client programs in the *CICS Transaction Gateway for Multiplatforms: Programming Guide* for further information.

Tracing in Java Applets

JEE Tracing

A detailed trace mechanism is provided for both the ECI and EPI resource adapters. Trace is useful when problem solving for applications that use the CICS resource adapters.

The CICS resource adapters support four levels of trace:

Level	Trace
0	No trace messages
1	Exception tracing only (default level)
2	Exception and method entry/exit trace messages
3	Exception, method entry/exit and debug trace messages

To provide more control over tracing, these system properties are available:

Property	Purpose
com.ibm.connector2.cics.tracelevel	Overrides the deployed trace level for the resource adapters without having to redeploy or deploy another CICS resource adapter.
com.ibm.connector2.cics.dumpoffset	The offset into a byte array at which a hex dump will start.
com.ibm.connector2.cics.dumplength	The maximum length of data displayed in a hex dump.
com.ibm.connector2.cics.outputerr	Declaring this directive sends trace output to standard error, if no other trace location has been specified either by the JEE server, or by the application developer working in a nonmanaged environment. In other circumstances the provided logwriter takes precedence.

These are JVM System properties that can be passed to the JVM on startup. The com.ibm.connector2.cics.tracelevel option is equivalent to the managed environment property "tracelevel" that is set as a custom property on the connection factory.

The connection manager thread used by your environment controls the location to where trace is written. In a managed environment an option should be provided to allow you to set the path of the file that will be used for storing trace. Depending on how your environment allocates connection manager threads to resource adapters this file might contain messages from other resource adapters using the same connection manager.

When you deploy the CICS resource adapters into your environment, security restrictions are set up to allow access to the local file system for the purpose of writing trace files.

Access is given to the IBM/ctg directory in your home directory.

This might map to:

```
/home/ctguser/IBM/ctg/
```

Therefore, when setting the name and path of the trace file in your JEE environment, use a location under this directory structure to store your trace. Otherwise the resource adapters will not have security permissions to write to the file.

Tracing issues when serializing Connection Factories

In a non-managed environment, when a ConnectionFactory object is serialized the reference to the LogWriter used for tracing is lost.

If you want trace to be written to a LogWriter you can use the setLogWriter method which can call on the DefaultConnectionManager object. This method ensures that the LogWriter is used on any Connection created from a ConnectionFactory, regardless of whether or not it was previously serialized and de-serialized. An example of this, writing trace to the standard error stream, is shown:

```
DefaultConnectionManager.setLogWriter(new java.io.PrintWriter(System.err));
Connection Conn = (Connection)cxf.getConnection();
```

The trace level within the ConnectionFactory is maintained throughout the serialization process and is unaffected by the LogWriter in the DefaultConnectionManager.

Problem solving and support

This section provides information about how to resolve problems with your IBM software, including instructions for searching knowledge databases, downloading fixes, and getting support.

IBM Technotes and other support documents are published on the CICS Transaction Gateway support Web site. You can also search Web-based support resources by using the customized query fields in the Web search topic. For more information, see *http://www-01.ibm.com/software/htp/cics/ctg/support/*.

Searching knowledge bases

If you have a problem with CICS Transaction Gateway, you want it resolved quickly. Begin by searching the available knowledge bases to determine whether the solution to your problem is already documented.

1. Search the CICS Transaction Gateway Information Center.
2. Search the Internet. If you cannot find an answer to your question in the information center, search the Internet for the latest, most complete information that might help you resolve your problem. To search multiple Internet resources for CICS Transaction Gateway, use the Web search tool. The tool enables you to search a variety of resources including:

   IBM Technotes

   Downloads

   IBM Redbooks publications

   IBM DeveloperWorks

   Forums and newsgroups

   Google

Contacting IBM Software Support

IBM Software Support provides assistance with product defects.

Before contacting IBM Software Support, your company must have an active IBM software maintenance contract, and you must be authorized to submit problems to IBM.

Follow the steps in this topic to contact IBM Software Support:

1. Determine the business impact of your problem.
2. Describe your problem and gather background information.
3. Submit your problem to IBM Software Support.

Determine the business impact of your problem

When you report a problem to IBM, you will be asked to supply a severity level. Therefore, you need to understand and assess the business impact of the problem you are reporting. Use the following criteria:

Severity	Impact	Characteristic
1	Critical	You are unable to use the program, resulting in a critical impact on operations. This condition requires an immediate solution.
2	Significant	The program is usable but is severely limited.
3	Moderate	The program is usable with less significant features (not critical to operations) unavailable.
4	Minimal	The problem causes little impact on operations, or a reasonable circumvention to the problem has been implemented.

Describe your problem and gather background information

When explaining a problem to IBM, be as specific as possible. Include all relevant background information so that IBM Software Support specialists can help you solve the problem efficiently. To save time, know the answers to these questions:

- What software versions were you running when the problem occurred?
- Do you have logs, traces, and messages that are related to the problem symptoms? IBM Software Support is likely to ask for this information.
- Can the problem be recreated? If so, what steps led to the failure?
- Have any changes been made to the system? For example, hardware, operating system, networking software, and so on.
- Are you currently using a workaround for this problem? If so, please be prepared to explain it when you report the problem.

To find out what information and files you will need to supply when opening a problem management record (PMR), see `http://www-01.ibm.com/support/docview.wss?uid=swg21287335#submit`

Submit your problem to IBM Software Support

You can submit your problem in one of two ways:

- Online: Go to the Submit and track problems page on the IBM Software Support site. Enter your information into the appropriate problem submission tool.
- By phone: For the phone number to call in your country, go to the contacts page of the IBM Software Support Handbook on the Web and click the name of your geographic region.

If the problem you submit is for a software defect or for missing or inaccurate documentation, IBM Software Support will create an Authorized Program Analysis Report (APAR). The APAR describes the problem in detail. Whenever possible, IBM Software Support will provide a workaround for you to implement until the APAR is resolved and a fix is delivered.

IBM publishes resolved APARs on the IBM product support Web pages daily, so that other users who experience the same problem can benefit from the same resolutions.

Chapter 13. Monitoring and statistics

Monitoring provides information about the status of individual requests. Statistics provide information about the performance of runtime components.

Monitoring

Request monitoring exits can optionally be used for driving user exit code on a per request basis. One or more user exit programs can be called for each request if details of each request are made available. All user exit code is called inline; this means that performance of the user exit code is critical.

Statistics

CICS Transaction Gateway statistics are always active and predefined by IBM. Unlike the request monitoring exits, the statistics provide summary information such as running totals, averages, status, and configuration values. Statistics are either displayed from system management commands, or can be obtained through a program that uses the statistics API.

Request monitoring exits

Request monitoring exits provide information about individual requests as they are processed by CICS Transaction Gateway.

Exit points in the product allow user code to be run in the context of each individual transaction. This context allows the user code to take action based on information specific to the current request. For example, an exit might be written to trigger an alert if an individual transaction runs for longer than a specified time. You use request monitoring exits to analyze transaction flows to assist with problem determination and performance tuning. Samples exits are provided; these demonstrate how request exits can be used.

The Java based request monitoring exits are available on the Gateway classes and the Gateway daemon and can be stacked, enabling multiple exits to be driven for an individual request. Exits are called for each ECI flow at the point of request entry to, and response exit from, the CICS Transaction Gateway code. The data available to the exit depends on the type of ECI flow and the point at which the exit is driven from.

The data values available to request monitoring exits are passed to the RequestExit eventFired() method.

Table 21. Data available to request monitoring exits

Request data description	Description
CicsAbendCode	CICS abend code on a response.
CicsReturnCode	CICS return code on a response.
CicsServer	The server to which CICS Transaction Gateway sent the request.
ClientCtgApplid	APPLID of the Client application.
ClientCtgApplidQualifier	APPLID qualifier of the Client application.
ClientCtgCorrelator	Correlator generated by the Java client application

Table 21. Data available to request monitoring exits (continued)

Request data description	Description
ClientLocation	Location of client Gateway classes (IP address).
CommandData	Command data originating from a request monitor exit administration request.
CtgApplid	CICS Transaction Gateway APPLID.
CtgApplidQualifier	CICS Transaction Gateway APPLID qualifier.
CtgCorrelator	CICS Transaction Gateway identifier used to track this flow within the CICS Transaction Gateway instance.
CtgReturnCode	CICS Transaction Gateway return code on a response.
DistributedIdentity	Distributed identity associated with this transaction.
FlowTopology	Topology from which the request exit was called: • gateway - from the Gateway daemon • remote - from a remote client • local - from a local client
FlowType	The flow type of this request or response.
GatewayUrl	URL of the Gateway to which the Java client is connecting.
Location	Location of this monitor. The value is an IP address.
LuwToken	CICS Transaction Gateway logical unit of work token.
OriginData	IPIC origin data. Use to identify the client that originated a CICS task. This is only available when communicating with CICS using the IPIC protocol.
PayLoad	A copy of the COMMAREA for use in the exit.
Program	CICS program name.
RequestReceived	Timestamp of request flow received in the Gateway classes or Gateway daemon classes; number of milliseconds since January 1, 1970, 00:00:00 GMT.
RequestSent	Timestamp of response flow sent from the Gateway classes or Gateway daemon classes; number of milliseconds since January 1, 1970, 00:00:00 GMT
ResponseReceived	Timestamp of response flow received in the Gateway classes or Gateway daemon classes; number of milliseconds since January 1, 1970, 00:00:00 GMT
ResponseSent	Timestamp of response flow sent from the Gateway classes or Gateway daemon classes; number of milliseconds since January 1, 1970, 00:00:00 GMT
RetryCount	The number of times the Gateway daemon retried sending a request to CICS.
Server	Server specified in the request.
TpnName	TPN Name.
TranName	Transaction ID.
Userid	User ID.
WireSize	Number of bytes of data received from or about to be sent to the Gateway classes.
WorkerWaitTime	Time in milliseconds that the Gateway daemon waited for a worker thread to become available to process the request. If the Gateway daemon times out waiting for a worker thread to become free, this value contains the time in milliseconds that the Gateway daemon waited before the timeout occurred.
XaReturnCode	XA return code on a response.
Xid	XID for XA transaction.

Related information:

Request monitoring exits configuration

Request monitoring exit API information

Request monitoring exits configuration

In a remote mode topology, you can configure request monitoring exits individually for the Gateway classes and the Gateway daemon. In a local mode topology, you must configure request monitoring exits for the Gateway classes.

In both situations, the exit configuration data must follow this format:

- Each exit must be defined using a fully qualified class name.
- Exits must be delimited from each other by commas (",").

When the Gateway classes or the Gateway daemon processes the configuration data, each class is instantiated and failures are logged. When a request monitoring exit object is used, any exceptions or runtime errors are logged and the exit becomes inactive.

For more information see "Configuring the request monitoring exits for a Gateway daemon" on page 102.

Configuring the request monitoring exits for a Gateway daemon

The **requestexits** parameter specifies a list of one or more classes that perform request monitoring.

Use the configuration tool to set the request monitors to use, or set the **requestexits** parameter to a valid class name for a request monitor class:

```
requestexits=fully_qualified_class_name
```

For example:

```
requestexits=com.ibm.ctg.samples.requestexit.MyMonitor
```

You can define multiple exits by separating them with a comma:

For example:

```
requestexits=com.ibm.ctg.samples.requestexit.1stMonitor,com.ibm.ctg.samples.requestexit.2ndMonitor
```

This parameter is in the GATEWAY section of the configuration file, see "GATEWAY section of the configuration file" on page 115 for more information about other parameters in this section.

Statistics

Statistics can help with problem determination and capacity planning, and make it possible to gain a snapshot of the current activity in CICS Transaction Gateway.

Statistics can be displayed using the administration interface or retrieved using a program through the Statistics API. Statistical values reflect the status or activity of the Gateway daemon and the Client daemon from which they were collected. Statistical data for the Gateway daemon is based on Client applications that run in remote mode. Statistical data for the Client daemon is based on Client applications that run in both local and remote modes and use a TCP/IP or SNA server connection. No statistical data is available for Java Client applications that run in local mode. The collection of statistics has an insignificant impact on performance, and statistics are always available.

CICS Transaction Gateway statistics aim to assist you in the following activities:

- Capacity planning information and throughput analysis
- Critical resource usage
- Problem determination

Interval and end-of-day statistics reflect those used by the CICS Transaction Server products to allow for synchronization of statistics collection between the products. Further information can be found from the Statistics parameters section of the *CICS Transaction Server for z/OS System Definition Guide*.

Resource group ID

A resource group ID is a logical grouping of resources, grouped for statistical purposes. A resource group ID is associated with a number of resource group statistics, each identified by a statistic ID.

Statistic ID

A statistic ID is a label referring to a specific statistical value, and is used to identify or retrieve statistical data. The statistic ID consists of three parts: <resource group ID>_<statistical type><statistic ID suffix>. For example, the statistic ID CM_CALLOC is part of the connection manager (CM) resource group, and represents the current (C) number of allocated (ALLOC) connection manager threads. See "List of statistics" on page 269 for a list statistic IDs arranged by resource group.

Statistical type

There are four statistical types:

C **Current**: the statistic is based on a current evaluation; the value is dynamic.

I **Interval**: the statistic is based on interval equivalents of existing Lifetime statistics, Gateway bandwidth or throughput, average response times, and thread use. The statistical values are reset periodically.

L **Lifetime**: the statistic is based on observations since the Gateway daemon started; the value is dynamic. Each lifetime statistic has a default value, which is set when the Gateway daemon is initialized.

S **Startup**: the statistic is based on a configuration setting for the Gateway daemon; the value is static.

If a characteristic of the product is reflected by a statistical ID of type Lifetime, in general there is an equivalent statistical ID of type Interval.

Statistic ID suffix

The statistic ID suffix is the part of the statistic ID that follows the statistical type character. This suffix is usually a noun representing the particular characteristic of the resource group represented by the statistic ID. Similar characteristics that are shared by resource groups can use the same statistic ID suffix for consistency. For example, the suffix ALLOC is used in statistical IDs WT_CALLOC, CM_CALLOC, and CS_CALLOC.

Resource groups

Statistics resource groups are a logical grouping of resources such as connection manager threads.

Statistics configuration

You can configure parameters for statistics interval, statistics end of day and statistics API port.

Interval timing patterns

The interval and end-of-day parameters combine to form a timing pattern. This timing pattern determines when statistics intervals begin and end. At interval boundaries statistics of statistical type "Interval" are reset to the default values. For examples, see “Interval timing patterns” on page 263.

Statistics interval

Statistics are gathered by CICS Transaction Gateway during a specified interval. You can change the interval value using the **statint** system parameter. You can set the **statint** parameter in these ways:

- The Gateway daemon panel in the Configuration Tool.
- The Gateway section of the configuration file.

The **statint** parameter equates to the CICS Transaction Server keyword and concepts of the parameter of the same name.

See “Statistics Interval (HHMMSS)” on page 105 for more information.

Statistics end of day

The end-of-day value (stateod) defines a logical point in the 24–hour operation of CICS Transaction Gateway.

You can change the end of day value:

- The Gateway daemon panel in the Configuration Tool. See “Statistics End of Day time (HHMMSS)” on page 105
- The Gateway section of the configuration file.

The **stateod** parameter equates to the CICS Transaction Server keyword and concepts of the parameter of the same name. See “Statistics End of Day time (HHMMSS)” on page 105 for more information.

Statistics API port

The Statistics API port allows the Gateway daemon to handle incoming requests for the Statistics API. You can select the port number on which to listen for Statistics API requests:

- The Gateway daemon panel in the Configuration Tool. See “Statistics API protocol settings” on page 103
- The Gateway section of the configuration file.

See “Statistics API protocol settings” on page 103 for more information.

Related reference:

"Statistics API protocol settings" on page 103
Use the CICS Transaction Gateway configuration tool to configure the statistics API protocol settings, or edit the statistics API protocol parameters in the GATEWAY section of the configuration file directly.

Related information:

"Statistics Interval (HHMMSS)" on page 105
The **statint** parameter specifies the recording interval for system statistics. The default is three hours. The interval must be at least one minute and cannot be more than 24 hours.

"Statistics End of Day time (HHMMSS)" on page 105
The **stateod** parameter specifies the end-of-day time. The End of Day time is used as a point of reference for the clock. Intervals are aligned to this rather than to the CICS Transaction Gateway startup time. This also determines the point at which statistics are reset and potentially recorded, and occurs at least once every 24 hours.

"GATEWAY section of the configuration file" on page 115
This table provides the names and descriptions for all parameters that can be set in the GATEWAY section of the configuration file.

Setting up your system for statistics

Use the configuration tool to configure your system to deal with requests for statistics.

Before you begin

Run the Configuration Tool and navigate to the **Resources** tab of the Gateway daemon node.

Procedure

1. Clear the **Disabled** check box.
2. Optional: enter a value in the **Statistics API port** if the default setting is not suitable.
3. Enter a value for the Statistics Interval in the format HHMMSS. The default value is three hours (030000).
4. Enter a value for the End-of-Day time in the format HHMMSS. The default value is midnight (000000).
5. Save the configuration file and then stop and restart the Gateway daemon.

Related reference:

"Statistics API protocol settings" on page 103
Use the CICS Transaction Gateway configuration tool to configure the statistics API protocol settings, or edit the statistics API protocol parameters in the GATEWAY section of the configuration file directly.

Related information:

"Statistics API protocol parameters" on page 117
To enable the statistics API protocol, include a protocol handler definition in the GATEWAY section of the configuration file.

Interval statistics

Two keywords, **statint** and **stateod**, are included in the GATEWAY section of the configuration file. The keyword **statint** shows the statistics interval duration and

the keyword **stateod** shows the End-of-Day time. These two keywords match the equivalent setting in CICS Transaction Server, and are familiar to CICS Transaction Server administrators.

If either or both keywords are undefined, on Gateway daemon initialization, they default to three hours **statint** and midnight **stateod**. The default is shown in the sample configuration file, ctgsamp.ini:

```
# Statistics interval duration specified in the form
# HHMMSS, where the overall range of valid interval
# values is from 000100 (1 minute) to 240000 (24 hours)
# inclusive. HH must be in the range 00 to 24, with MM
# and SS values in the range 00 to 59.
statint=030000

# Statistics end of day time in the form HHMMSS, and is
# expressed in local time, where the overall range of
# valid time is from 000000 (midnight) to 235959 (1 second
# before midnight) inclusive. HH must be in the range 00
# to 23, with MM and SS values in the range 00 to 59.
stateod=000000
```

The two fields **Statistics interval** and **Statistics End of Day time** are shown in the Gateway daemon panel in the Configuration Tool. The fields are located below the Statistics API port field.

The **Statistics Interval** combines with the **Statistics End of Day time** to formulate times at which interval statistics are reset. Reset occurs at the end of the current interval or at the **Statistics End of Day time** (the logical end of day), whichever comes first. Valid values for the statistics interval parameter, **statint**, are between 1 minute and 24 hours. The field requires the interval to be specified in the format HHMMSS, and accepts interval times only within the specified range.

If an irregular interval is specified and the end of interval and the **Statistics End of Day time** might not coincide, that interval is truncated. The next interval starts from **Statistics End of Day time**. For further details see "Interval timing patterns." Valid values for the End of Day time parameter, **stateod**, can range between midnight (000000) and 1 second before midnight (235959). The field requires the interval to be specified in the format HHMMSS, and accepts interval time only within the specified range.

Related information:

"GATEWAY section of the configuration file" on page 115
This table provides the names and descriptions for all parameters that can be set in the GATEWAY section of the configuration file.

Interval timing patterns

Interval boundaries are aligned to the logical end of day. It is most likely that the first statistics interval after Gateway daemon initialization will be of a shorter duration than the configured interval length.

The first interval period is shorter than subsequent interval periods if either of the following conditions are met:

- The Gateway daemon start time does not match one of the interval end times, as shown in the examples in the following tables.
- You select a figure for an interval period that does not divide equally into 24, for example 5 hours (050000).

Examples of Statistics Interval timings

Interval Statistics timing – Example 1:

The Gateway daemon starts at 5:20 a.m. (052000) and is configured with **statint=030000 stateod=000000**.

The first Interval is scheduled to end at 6:00 a.m. (060000), and is of 40 minutes duration. This schedule allows subsequent intervals be aligned with the end-of-day event. In this case, statistics are reset and optionally recorded at the following times during the 27 hours following Gateway initialization:

Table 22. Interval Statistics timing – Example 1

Time	Event type	Interval length HH:MM:SS
05:20:00	Gateway starts	Not applicable
06:00:00	Interval reset	00:40:00
09:00:00	Interval reset	03:00:00
12:00:00	Interval reset	03:00:00
15:00:00	Interval reset	03:00:00
18:00:00	Interval reset	03:00:00
21:00:00	Interval reset	03:00:00
00:00:00	End of Day reset	03:00:00
03:00:00	Interval reset	03:00:00
Sequence repeats		

Interval Statistics timing – Example 2:

The Gateway daemon starts at 5:20 a.m. (052000) and is configured for six hour statistics intervals, with logical end of day at 23:59 with **statint=060000 stateod=235900**.

The first Interval is scheduled to end at 5:59 a.m. (055900), and is of 39 minutes duration. This schedule allows subsequent intervals be aligned with the end-of-day event. In this case, statistics are reset and optionally recorded at the following times during the 30 hours following Gateway initialization:

Table 23. Interval Statistics timing – Example 2

Time	Event type	Interval length HH:MM:SS
05:20:00	Gateway starts	Not applicable
05:59:00	Interval reset	00:39:00
11:59:00	Interval reset	06:00:00
17:59:00	Interval reset	06:00:00
23:59:00	End of Day reset	06:00:00
05:59:00	Interval reset	06:00:00
11:59:00	Interval reset	06:00:00
Sequence repeats		

Interval Statistics timing - Example 3:

The Gateway daemon starts at 5:20 a.m. (052000) and is configured for a 24 hour statistics interval, with logical end of day at 23:59 with **statint=240000 stateod=235900**.

The first Interval is scheduled to end at 23:59 (235900), and is of 17 hours and 39 minutes duration. This schedule allows subsequent intervals be aligned with the end-of-day event. In this case, statistics are reset and optionally recorded at the following times during the days following Gateway initialization:

Table 24. Interval Statistics timing – Example 3

Time	Event type	Interval length HH:MM:SS
05:20:00	Gateway starts	Not applicable
23:59:00	End of Day reset	17:39:00
23:59:00	End of Day reset	24:00:00
23:59:00	End of Day reset	24:00:00
Sequence repeats		

Displaying statistics

You can use the **ctgadmin** command to display statistical information about the CICS Transaction Gateway, or obtain statistics using either the C or Java Statistics API interface.

Use ctgadmin to display statistical information about the CICS Transaction Gateway. Use the options listed in Statistical options to display statistical information. Do not combine options.

Enter a command like the following at the command line:

```
ctgadmin -a stats options
```

For example, to display all available statistics about the CICS Transaction Gateway, enter the following command:

```
ctgadmin -a stats -gs
```

The command is not case-sensitive.

Displaying all available statistics

Use ctgadmin with the -gs option with no parameters, to display all available statistical information about the CICS Transaction Gateway.

Enter the following command:

- `ctgadmin -a stats -gs`.

Selecting the statistics to display

Use ctgadmin, with the -gs option followed by a list of IDs, to display statistics for one or more statistical IDs and resource groups.

Use the -gs option, followed by a list of IDs for the statistics or resource groups. Separate each item in the list by a colon (:).

For example, to display information about the worker thread resource group, the maximum number of connection managers, and the Gateway daemon resource group, enter the following command:

```
ctgadmin -a stats -gs wt:cm_smax:gd
```

You might use an optional parameter, **stattype**, (st) to filter on statistic type. The parameter is case-insensitive and consists of colon-separated single characters each of which denotes a statistic type:

- S = Startup
- C = Current
- L = Lifetime
- I = Interval

To display the interval statistical values, for all resource groups, enter one of the following commands:

```
ctgadmin –a stats –gs –st=i
```

```
ctgadmin –a stats –getstats –stattype=i
```

To display the current statistical values from the CS resource group, enter the following command:

```
ctgadmin –a stats –gs=cs –st=c
```

To display all lifetime and interval statistical values from the GD resource group, enter the following command:

```
ctgadmin –a stats –gs=gd –st=L:I
```

If the **stattype** option is omitted, the output is unfiltered. Any repeated statistical type characters or unrecognized statistical type characters are ignored. If any of the specified statistical type characters are unrecognized, the command produces a warning message.

Listing available resource groups

Use ctgadmin, together with the -rg parameter with no other options, to list available resource groups.

To list available resource groups, enter the following command:

- `ctgadmin -a stats -rg`.

Listing all available statistical IDs

Use ctgadmin, together with the -si parameter, to list available statistical IDs.

To list all available statistical IDs, enter the following command:

- `ctgadmin -a stats -si`.

Listing statistical IDs for selected resource groups

Use ctgadmin, with the -si parameter followed by a list IDs, to list available statistical IDs.

To list statistical IDs for one or more resource groups, use the -si parameter followed by a colon-separated list of resource groups. For example, to list statistical IDs for the connection manager and worker thread resource groups, enter the following command:

- `ctgadmin -a stats -si cm:wt`.

Getting help on statistics

Use ctgadmin -a stats -? to get help on statistics.

Before you begin

Issue the following command:

Procedure

ctgadmin -a stats -?

Statistics resource groups

Every statistic belongs to a *resource group*. Resource groups define an area for which statistical data can be associated and retrieved. Resource groups are available from the CICS Transaction Gateway.

A resource group is a logical grouping of resources, such as connection managers. It defines an area for which statistical data can be associated and retrieved. Each resource group has these characteristics:

ID A unique identifier for the resource group. The ID is used by ctgadmin and the statistical API to retrieve statistics, and is not case-sensitive.

Name
The name of the resource group, displayed when ctgadmin is used to display statistical information.

Description
A description of the resource group.

These resource groups are defined:

Table 25. Resource groups

ID	Name	Description
CD	Client daemon statistics	Statistics about the Client daemon process.
CM	Connection manager statistics	Statistics about connection manager threads.
CS	CICS server (all) statistics	Statistics about all CICS servers.
CS*x*	CICS server (instance) statistics	Statistics for an individual CICS server, where *x* is the APPLID of the CICS server.
GD	Gateway daemon statistics	Statistics on transaction counts, request counts, and Gateway status.
PH	Protocol handler statistics	Statistics about protocol handlers.
SE	System environment statistics	Statistics about the System Environment of the Gateway daemon.
WT	Worker thread statistics	Statistics about worker threads.

Client daemon resource group (CD)

The Client daemon resource group contains statistical values reflecting the status and activity of the Client daemon component. Client daemon statistics are only available through the Gateway daemon administration interface, ctgadmin, or the statistics API. This means that both the Gateway daemon and the Client daemon must be active to display or retrieve Client daemon statistics.

Client daemon statistics are initialized when the Client daemon is started and the values are not reset if the Gateway daemon is started later. If a CICS server connection is stopped using the CICSCLI command, then statistics for that CICS server will persist for the lifetime of the Client daemon. If a CICS server connection is stopped and restarted the statistic values are not reset.

Connection manager resource group (CM)

Statistics are available for connection manager threads. These statistics identify the characteristics for the pool of connection manager threads and are useful for analyzing resource usage, capacity planning and diagnosing system problems.

CICS Server (all) resource group (CS)

Statistics are available that summarize interactions with all associated CICS servers.

CICS Server (instance) resource group (CS*x*)

Statistics are available for each specific CICS server "x". In general, the statistic IDs of the CS resource group which summarize activity across all associated CICS servers, can be found for each CICS server in the corresponding CS*x* resource group.

For example, a CICS Transaction Gateway connected to a CICS server defined by the name *CICSAOR1* is represented by resource group *CSCICSAOR1*. An example of a statistical ID available for such a resource group is *CSCICSAOR1_LALLREQ*. If the server name contains any underscores (_), they are replaced with hyphens (-) in the resource group ID.

A CS*x* resource group is available for a connected CICS server regardless of the protocol used. However, there are some protocol-specific statistic IDs. The CS*x*_SPROTOCOL statistic is provided to distinguish the set of values that can be expected for a given CS*x* resource group, especially for use by a Statistics API program.

Statistics for CICS servers connected over SNA and TCP/IP are available immediately after initialization. Statistics for CICS servers connected over IPIC are available after a connection is attempted.

Gateway daemon resource group (GD)

Statistics are available for the Gateway daemon. These statistics include status, an indication of work done on behalf of remote mode Client applications, completed and active transactions. Although there is a correlation between the number of requests counted in the GD and CS resource groups, requests counted in the GD resource group reflect remote Client requests and are likely to be different from the CS resource group request counts. Some Client requests do not require any interaction with a CICS server; for example, list systems. Other Client requests might require more than one interaction with a CICS server.

System environment resource group (SE)

Statistics are available to assist in the analysis of storage usage by the Gateway daemon. JVM Heap storage and Garbage Collection (GC) information is provided.

Worker thread resource group (WT)

Statistics are available for the worker threads. These statistics identify the characteristics for the pool of worker threads that can be used by connection managers. These statistics are useful for analyzing resource usage and capacity planning, and for diagnosing system problems.

List of statistics

These statistics are available from the CICS Transaction Gateway.

Each statistic has the following characteristics:

ID A unique identifier for the statistic. The ID is used by ctgadmin and the statistical API to retrieve statistics, and is not case-sensitive. The structure of the ID is as follows:

```
<resource group>_<statistics type><statistics suffix>
```

Each part of the ID is mandatory; their characteristics are as follows:

<resource group>
An alphanumeric string of one or more characters representing the resource group to which the statistic belongs.

<statistics type>
A single character; valid values are C, I, L, and S.

- **C** **Current**: the statistic is based on a current evaluation; the value is dynamic.
- **I** **Interval**: the statistic is based on interval equivalents of existing Lifetime statistics, Gateway bandwidth or throughput, average response times, and thread utilization.
- **L** **Lifetime**: the statistic is based on observations since the Gateway daemon started; the value is dynamic. Each lifetime statistic has a default value, which is set when the Gateway daemon is initialized.
- **S** **Startup**: the statistic is based on a configuration setting for the Gateway daemon; the value is static.

<statistics suffix>
An alphanumeric string of one or more characters representing the resource about which information is being returned.

Short description
The short description is displayed when ctgadmin is used to display statistical information.

Description
A description of the information returned by the statistic.

Value returned
The type of information returned by the statistics:

- **Integer** The string value represents a 4-byte numeric value.
- **Long** The string value represents an 8-byte numeric value.
- **String** The string value represents character data.

These subtopics describe the statistics that are defined:

Connection manager statistics:

The statistics listed here belong to the connection manager resource group.

Table 26. Connection manager statistics

ID	Description	Default value	Data type
CM_CALLOC	The current number of connection manager threads allocated to clients.	0	Integer
CM_CCURR	The current number of connection manager threads created.	0	Integer
CM_CWAITING	The current number of connection managers waiting for a worker thread to become available.	0	Integer
CM_IALLOC	The number of allocations for connection manager threads representing the number of connections that have been established from remote clients. A low value represents efficient connection reuse.	0	Integer
CM_IALLOCHI	The peak number of connection manager threads concurrently allocated to client applications. This number represents a high water mark for CM_CALLOC.	<CM_CALLOC>	Integer
CM_ICREATED	The number of connection manager threads created.	0	Integer
CM_ITIMEOUTS	The number of times that the Gateway daemon failed to allocate a connection manager thread to a client application within the defined **connecttimeout** length of time.	0	Integer
CM_LALLOC	The number of allocations for connection manager threads representing the number of connections that have been established from remote clients. A stable value represents efficient connection reuse.	0	Integer
CM_LTIMEOUTS	The number of times that the Gateway daemon failed to allocate a connection manager thread to a client application within the defined (connecttimeout) length of time.	0	Integer
CM_SINIT	The initial number of connection manager threads **initconnect** created by the Gateway daemon.	0	Integer
CM_SMAX	The maximum number of connection manager threads **maxconnect** that can possibly be created and allocated by the Gateway daemon.	0	Integer **Note:** A value of -1 indicates no limit.

CICS server (all) statistics:

The statistics listed here belong to the CICS server (all) resource group.

Table 27. CICS server (all) statistics

ID	Description	Default value	Data type
CS_CORPHANREQ	The number of requests currently waiting for a response from CICS for which the owning application has timed out or ended.	0	Integer
CS_CREQCURR	The current number of active requests in the Client daemon. The maximum number of active requests is defined by CS_SREQMAX	0	Integer
CS_CSESSCURR	The number of IPIC sessions in use with CICS servers.	0	Integer
CS_CSESSMAX	The number of IPIC sessions negotiated with CICS servers.	0	Integer
CS_CTERM	The current number of installed terminals including EPI, cicsterm and cicsprnt.	0	Integer
CS_CWAITING	The number of requests currently waiting for a response from a CICS server.	0	Integer
CS_ISESSFAIL	The number of failures on IPIC sessions to CICS servers.	0	Integer
CS_LALLREQ	The number of requests to CICS servers (successful and failed) that have been processed.	0	Integer
CS_LAVRESP	The average time taken (in milliseconds) for a connected CICS server to respond to the Gateway daemon over the lifetime of the current Gateway daemon.	0	Integer
CS_LCONNFAIL	The number of times an attempt to connect to a CICS server has failed.	0	Integer
CS_LCOUNT	The number of CICS servers to which requests have been sent. This number equals the number of CICS servers in CS_LLIST.	0	Integer
CS_LIDLETIMEOUT	The number of times a connection to a CICS server has timed out.	0	Integer
CS_LLIST	The list of CICS servers to which requests have been sent.	<empty string>	String
CS_LLOSTCONN	The number of times an established connection with a CICS server has been lost.	0	Integer
CS_LORPHANREQ	The number of requests that have had to wait for a response from CICS for which the owning application has timed out or ended.	0	Integer
CS_LRESPDATA	The amount of response data (in bytes) received from connected CICS servers. This amount includes both application and CICS protocol data. For Client daemon and IPIC, the data comprises COMMAREA and CICS headers.	0	Long
CS_LREQDATA	The amount of request data (in bytes) sent to connected CICS servers. This amount includes both application and CICS protocol data. For Client daemon and IPIC, the data comprises COMMAREA and CICS headers.	0	Long
CS_LSESSFAIL	The number of failures on IPIC sessions to CICS servers.	0	Integer

Table 27. CICS server (all) statistics (continued)

ID	Description	Default value	Data type
CS_LTERMINST	The number of terminal installs processed by the Client daemon including EPI, cicsterm and cicsprnt. Successful and failed requests are included.	0	Integer
CS_LTERMUNINST	The number of terminals uninstalled by the Client daemon including EPI, cicsterm and cicsprnt. An uninstall event might be triggered by an application request or cleanup processing.	0	Integer
CS_SCOUNT	The number of CICS servers defined in the configuration file.	0	Integer
CS_SLIST	The list of all CICS servers defined in the configuration file.	<empty string>	String
CS_SREQMAX	The defined maximum number of active requests in the Client daemon. For the client this value is the same as the configuration file parameter MAXREQUEST	0	Integer

CICS server (instance) statistics:

The statistics listed here belong to the CICS server (instance) resource group.

Table 28. CICS server (instance) statistics

ID	Description	Default value	Data type
CS*x*_CAPPLID	The APPLID of the connected CICS server *x*.	<empty string>	String
CS*x*_CAPPLIDQ	The APPLID qualifier of the connected CICS server *x*.	<empty string>	String
CS*x*_CORPHANREQ	The number of requests currently waiting for a response from CICS server *x* for which the owning application has timed out or ended.	0	Integer
CS*x*_CREQCURR	The current number of active requests to CICS server *x*.	0	Integer
CS*x*_CSESSCURR	The number of IPIC sessions in use with CICS server *x*.	0	Integer
CS*x*_CSESSMAX	The number of IPIC sessions negotiated with CICS server *x*.	0	Integer
CS*x*_CTERM	The current number of terminals installed to CICS server *x* including EPI, cicsterm and cicsprnt.	0	Integer
CS*x*_CWAITING	The number of requests currently waiting for a response from CICS server *x*.	0	Integer
CS*x*_ISESSFAIL	The number of failures on IPIC sessions to CICS server *x*.	0	Integer
CS*x*_LALLREQ	The number of requests to CICS server *x* (successful and failed) that have been processed.	0	Integer
CS*x*_LAVRESP	The average time taken (in milliseconds) for connected CICS server x to respond.	0	Integer

Table 28. CICS server (instance) statistics (continued)

ID	Description	Default value	Data type
CS*x*_LCOMMSFAIL	The number of times communication with CICS server *x* has failed after a connection has already been established, or when there has been a failure with the link during communication with the server.	0	Integer
CS*x*_LCONNFAIL	The number of times an attempt to connect to a CICS server has failed.	0	Integer
CS*x*_LIDLETIMEOUT	The number of times a connection to CICS server *x* has timed out.	0	Integer
CS*x*_LLOSTCONN	The number of times an established connection with a CICS server has been lost.	0	Integer
CS*x*_LORPHANREQ	The number of requests that have had to wait for a response from CICS server *x* for which the owning application has timed out or ended.	0	Integer
CS*x*_LREQDATA	The amount of request data (in bytes) sent to connected CICS server *x*. This amount includes both application and CICS protocol data. For Client daemon and IPIC, the data comprises COMMAREA and CICS headers.	0	Long
CS*x*_LRESPDATA	The amount of response data (in bytes) received from connected CICS server *x*. This amount includes both application and CICS protocol data. For Client daemon and IPIC, the data comprises COMMAREA and CICS headers.	0	Long
CS*x*_LSESSFAIL	The number of failures on IPIC sessions to CICS server *x*.	0	Integer
CS*x*_LTERMINST	The number of terminal installs processed for CICS server *x* including EPI, cicsterm and cicsprnt. Successful and failed requests are included.	0	Integer
CS*x*_LTERMUNINST	The number of terminals uninstalled from CICS server *x* by the Client daemon including EPI, cicsterm and cicsprnt. An uninstall event might be triggered by an application request or cleanup processing.	0	Integer
CS*x*_SIPADDR	The defined host name or IP address of the CICS server.	<empty string>	String
CS*x*_SIPPORT	The TCP/IP port of the CICS server.	0	Integer
CS*x*_SMODE	The defined SNA mode name of the LU 6.2 connection to the CICS server.	<empty string>	String
CS*x*_SNETNAME	The defined SNA partner LU 6.2 name for the CICS server. This name is a fully qualified LU name or an LU alias.	<empty string>	String
CS*x*_SPROTOCOL	The protocol used to communicate with the CICS server *x*. The protocol name is one of the following: IPIC, SNA, TCPIP.	N/A	String
CS*x*_SSESSMAX	The number of requested IPIC sessions for CICS server *x*.	0	Integer

Gateway daemon statistics:

The statistics listed here are members of the Gateway daemon resource group.

Table 29. Gateway daemon statistics

ID	Description	Default value	Data type
GD_CLUWTXN	The current number of inflight extended LUW transactions. These transactions might or might not be active in a CICS server; however, they always represent one mirror transaction, which might be in a suspended state.	0	Integer
GD_CNEXTRESET	The local time of the next scheduled interval statistics reset event (and optionally recording event). The value is in 24-hour HHMMSS format.	First scheduled reset time.	String
GD_CSTATUS	The status of the Gateway daemon. Status is one of the following: STARTING, RUNNING, SHUTTING DOWN.	N/A	String
GD_CSYNCTXN	The current number of inflight SYNCONRETURN transactions.	0	Integer
GD_IALLREQ	The number of API calls (ECI, ESI, EPI) and XA requests that have been processed. Successful and failed requests are included. Administrative requests and handshakes are excluded.	0	Integer
GD_IAVRESP	The average time taken in milliseconds for the Gateway daemon to respond to API (ECI, ESI, EPI) and XA requests from remote clients. Successful and failed requests are included. This value is inclusive of the CICS response time, as provided by the corresponding CS_IAVRESP statistic.	0	Integer
GD_IAVRESPIO	The average time in milliseconds for the Gateway daemon to respond to API (ECI) and XA requests from remote clients including network I/O time. Successful and failed requests are included. This value is inclusive of the Gateway response time, as provided by the corresponding GD_IAVRESP statistic.	0	Integer
GD_IHAEXIT	The number of times the CICS request exit was called.	0	Integer
GD_ILUWTXNC	The number of extended LUW-based transactions that were committed. This statistic returns information about 1–phase commit transactions.	0	Integer
GD_ILUWTXNR	The number of extended LUW-based transactions that were rolled back. This statistic returns information about 1–phase commit transactions.	0	Integer
GD_IREQDATA	The amount of request data in bytes received from client applications. All requests are included.	0	Long

Table 29. Gateway daemon statistics (continued)

ID	Description	Default value	Data type
GD_IRESPDATA	The amount of response data in bytes sent to client applications. All responses are included.	0	Long
GD_IRUNTIME	The time in seconds since the last reset event, or age of the current interval.	0	Integer
GD_ISYNCFAIL	The number of SYNCONRETURN transactions that have failed in the current interval.	0	Integer
GD_ISYNCTXN	The number of successful SYNCONRETURN transactions.	0	Integer
GD_LALLREQ	The number of API calls (ECI, ESI, EPI) and XA requests that have been processed. Successful and failed requests are included. Administrative requests and handshakes are excluded.	0	Integer
GD_LAVRESP	The average time in milliseconds for the Gateway daemon to respond to API (ECI, ESI, EPI) and XA requests from remote clients. Successful and failed requests are included. This value is inclusive of the CICS response time, as provided by the corresponding CS_LAVRESP statistic.	0	Integer
GD_LAVRESPIO	The average time in milliseconds for the Gateway daemon to respond to API (ECI) and XA requests from remote clients including network I/O time. Successful and failed requests are included. This value is inclusive of the Gateway response time, as provided by the corresponding GD_LAVRESP statistic.	0	Integer
GD_LHAEXIT	The number of times the CICS request exit was called.	0	Integer
GD_LLUWTXNC	The number of extended LUW-based transactions that were committed. This statistic returns information about 1–phase commit transactions.	0	Integer
GD_LLUWTXNR	The number of extended LUW-based transactions that were rolled back. This statistic returns information about 1–phase commit transactions.	0	Integer
GD_LREQDATA	The amount of request data in bytes received from client applications. All requests are included.	0	Long
GD_LRESPDATA	The amount of response data in bytes sent to client applications. All responses are included.	0	Long
GD_LRUNTIME	The length of time in seconds since the Gateway daemon successfully initialized.	0	Long
GD_LSYNCFAIL	The number of SYNCONRETURN transactions that have failed for the duration of the Gateway daemon process.	0	Integer

Table 29. Gateway daemon statistics (continued)

ID	Description	Default value	Data type
GD_LSYNCTXN	The number of successful SYNCONRETURN transactions.	0	Integer
GD_SAPPLID	The APPLID of the CICS Transaction Gateway, which identifies the instance of the CICS Transaction Gateway on CICS server connections.	<empty string>	String
GD_SAPPLIDQ	The APPLID qualifier of the CICS Transaction Gateway. GD_SAPPLIDQ is used as a high-level qualifier for the APPLID of the CICS Transaction Gateway. In combination with the APPLID, the fully qualified APPLID identifies the Gateway to the CICS system to which it connects.	<empty string>	String
GD_SHOSTNAME	The host name of the CICS Transaction Gateway computer. If the host name cannot be determined this statistic is set to "Unknown".	N/A	String
GD_SPLATFORM	The platform on which the CICS Transaction Gateway is running. Platform is one of the following: AIX, HP-UX (Itanium), Linux (Intel), Linux (POWER), Linux (zSeries), Solaris, Windows, z/OS, Unknown.	"Unknown"	String
GD_SDFLTSRV	The default CICS server for the CICS Transaction Gateway.	N/A	String
GD_SSTATEOD	The local time to be designated as the logical end of day by a Gateway daemon. At the logical end-of-day, all interval statistics are reset according to their defined default value. If the **statint** parameter has been set to an irregular value, the interval immediately prior to the **stateod** end-of-day is truncated. The value is in 24-hour HHMMSS format.	The value of the stateod parameter in the configuration file. If not specified in the configuration file the default value will be midnight, local time.	String
GD_SSTATINT	The duration of the statistics interval in use by a Gateway daemon. At the end of each interval, all interval statistics are reset according to their defined default value. The value is in HHMMSS format.	The value of the statint parameter in the configuration file. If not specified in the configuration file the default value represents 3 hours.	String
GD_SVER	The version of the CICS Transaction Gateway.	N/A	String

Client daemon statistics:

The statistics listed here belong to the Client daemon resource group.

Table 30. Client daemon statistics

ID	Description	Default value	Data type
CD_CAPPCURR	The current number of client application processes connected to the Client daemon. The Gateway daemon counts as one application.	0	Integer
CD_CSTATUS	The status of the Client daemon. Status is one of the following: STARTING, RUNNING, SHUTTING DOWN, SHUT DOWN.	N/A	String
CD_LALLREQ	The number of API calls (ECI, EPI, ESI) requests that have been processed. Successful and failed requests are included. Administrative requests are excluded. The Client daemon counts API list system requests and CICS server requests.	0	Integer
CD_LRUNTIME	The length of time (in seconds) since the Client daemon successfully initialized.	0	Long

System environment statistics:

The statistics listed here belong to the System Environment resource group.

Table 31. System Environment statistics

ID	Description	Default value	Data type
SE_CHEAPGCMIN	The Gateway daemon JVM heap size (in bytes) after the last garbage collection (GC).	0	Long
SE_IGCCOUNT	The number of garbage collection (GC) events.	0	Long
SE_IGCTIME	The length of time (in milliseconds) taken by the JVM for garbage collection (GC).	0	Long
SE_LGCCOUNT	The number of garbage collection (GC) events.	0	Long
SE_LGCTIME	The length of time (in milliseconds) taken by the JVM for garbage collection (GC).	0	Long

Table 31. System Environment statistics (continued)

ID	Description	Default value	Data type
SE_SHEAPINIT	The size of the Gateway daemon initial JVM heap (in bytes).	0	Long
SE_SHEAPMAX	The size of the Gateway daemon maximum JVM heap (in bytes).	0	Long

Protocol handler statistics:

The protocol handler bind and port statistics listed here belong to the Protocol handler resource group.

Table 32. Protocol handler statistics

ID	Description	Default value	Data type
PH_SBINDSSL	The address or host name to which the SSL protocol handler is bound. This statistic does not contain a value if the protocol handler is not enabled.	none	String
PH_SBINDTCP	The address or host name to which the TCP protocol handler is bound. This statistic does not contain a value if the protocol handler is not enabled.	none	String
PH_SPORTSSL	The SSL protocol handler port number, or -1 if the protocol is not enabled.	-1	Integer
PH_SPORTTCP	The TCP protocol handler port number, or -1 if the protocol is not enabled.	-1	Integer

Worker thread statistics:

The statistics listed here belong to the worker thread resource group.

Table 33. Worker thread statistics

ID	Description	Default value	Data type
WT_CALLOC	The current number of worker threads that are being used by connection managers. Another way of viewing this value is the number of worker threads processing requests.	0	Integer
WT_CCURR	The current number of worker threads created.	0	Integer
WT_IALLOCHI	The peak number of worker threads concurrently allocated to connection manager threads. This number represents a high water mark for WT_CALLOC.	<WT_CALLOC>	Integer
WT_ITIMEOUTS	The number of times the Gateway daemon failed to allocate a worker thread to a connection manager within the defined **workertimeout** length of time.	0	Integer

Table 33. Worker thread statistics (continued)

ID	Description	Default value	Data type
WT_LTIMEOUTS	The number of times the Gateway daemon failed to allocate a worker thread to a connection manager within the defined **workertimeout** length of time.	0	Integer
WT_SINIT	The initial number of worker threads **initworker** created by the Gateway daemon.	0	Integer
WT_SMAX	The maximum number of parallel requests **maxworker** that the Gateway daemon can process.	0	Integer A value of -1 indicates no limit.

Using the statistics

This information classifies statistics into different categories, according to how they are most likely to be used. Some statistics are in more than one category.

Statistics for tuning and capacity planning

Look at these key statistics when analyzing the performance of the CICS Transaction Gateway.

Capture statistics when the CICS Transaction Gateway is operating under a number of different operating conditions. This will help you understand changes that might affect the performance of the system.

CM_CALLOC
The current number of connection manager threads allocated to clients.

CM_CCURR
The current number of connection manager threads created. If this value is greater than the configuration parameter **initconnect**, it signifies the peak number of remote clients connected at any one time. This value cannot exceed the maximum number of connection managers defined in CM_SMAX.

CM_CWAITING
The current number of connection managers waiting for a worker thread to become available. This statistic shows the number of requests that are queuing in the Gateway daemon. It is usually low or zero in a well-tuned Gateway daemon. If it is higher than expected, consider increasing the **maxworker** configuration parameter.

CM_IALLOC
The number of allocations for connection manager threads representing the number of connections that have been established from remote clients. A low value represents efficient connection reuse.

CM_IALLOCHI
The peak number of connection manager threads concurrently allocated to client applications. This number represents a high water mark for CM_CALLOC.

CM_ICREATED
The number of connection manager threads created.

CM_ITIMEOUTS
The number of times that the Gateway daemon failed to allocate a connection manager thread to a client application within the defined **connecttimeout** length of time.

CM_LALLOC
The number of allocations for connection manager threads representing the number of connections that have been established from remote clients. A stable value represents efficient connection reuse.

CM_LTIMEOUTS
The number of times that the Gateway daemon failed to allocate a connection manager thread to a client application within the defined (connecttimeout) length of time. This statistic shows the number of incoming connection requests that have been refused. It is usually low or zero in a well-tuned Gateway daemon. If it is high, consider increasing the **connecttimeout** or **maxconnect** configuration parameters.

CM_SMAX
The maximum number of connection manager threads **maxconnect** that can possibly be created and allocated by the Gateway daemon. This value limits the number of Java clients that can be connected at any one time.

WT_CALLOC
The current number of worker threads that are being used by connection managers. Another way of viewing this value is the number of worker threads processing requests. If this value is close to WT_SMAX, consider increasing the **maxworker** configuration parameter.

WT_CCURR
The current number of worker threads created. If this value is greater than the configuration parameter **initworker**, it signifies the peak number of parallel requests that have been in process at any one time. This value cannot exceed the maximum number of worker threads defined in WT_SMAX.

WT_IALLOCHI
The peak number of worker threads concurrently allocated to connection manager threads. This number represents a high water mark for WT_CALLOC.

WT_ITIMEOUTS
The number of times the Gateway daemon failed to allocate a worker thread to a connection manager within the defined **workertimeout** length of time.

WT_LTIMEOUTS
The number of times the Gateway daemon failed to allocate a worker thread to a connection manager within the defined **workertimeout** length of time. This number signifies that requests are timing out while queuing in the Gateway daemon. It is typically low or zero in a well-tuned Gateway daemon. If it is higher than expected, consider increasing the **maxworker** or **workertimeout** configuration parameters.

WT_SMAX
The maximum number of parallel requests **maxworker** that the Gateway daemon can process.

Statistics for diagnosing system problems
Look at these key statistics when diagnosing system problems.

CM_ITIMEOUTS and CM_LTIMEOUTS
The number of times that the Gateway daemon failed to allocate a connection manager thread to a client application within the defined **connecttimeout** length of time.

GD_IAVRESP and GD_LAVRESP
The average time taken in milliseconds for the Gateway daemon to respond to API (ECI, ESI, EPI) and XA requests from remote clients. Successful and failed requests are included. This value is inclusive of the CICS response time, as provided by the corresponding CS_IAVRESP statistic.

WT_ITIMEOUTS and WT_LTIMEOUTS
The number of times the Gateway daemon failed to allocate a worker thread to a connection manager within the defined **workertimeout** length of time.

Statistics for the analysis of resource usage

Look at these key statistics when considering the resources used by your system.

CM_CALLOC
The current number of connection manager threads allocated to clients.

CM_CCURR
The current number of connection manager threads created. If this value is greater than the configuration parameter **initconnect**, it signifies the peak number of remote clients connected at any one time. This value cannot exceed the maximum number of connection managers defined in CM_SMAX.

CM_IALLOC
The number of allocations for connection manager threads representing the number of connections that have been established from remote clients. A low value represents efficient connection reuse.

CM_IALLOCHI
The peak number of connection manager threads concurrently allocated to client applications. This number represents a high water mark for CM_CALLOC.

CM_ICREATED
The number of connection manager threads created.

CM_LALLOC
The number of allocations for connection manager threads representing the number of connections that have been established from remote clients. A stable value represents efficient connection reuse.

CS_LCOUNT
The number of CICS servers to which requests have been sent. This number equals the number of CICS servers in CS_LLIST.

CS_LLIST
The list of CICS servers to which requests have been sent.

WT_CALLOC
The current number of worker threads that are being used by connection managers. Another way of viewing this value is the number of worker threads processing requests. If this value is close to WT_SMAX, consider increasing the **maxworker** configuration parameter.

WT_CCURR
The current number of worker threads created. If this value is greater than the configuration parameter **initworker**, it signifies the peak number of parallel

requests that have been in process at any one time. This value cannot exceed the maximum number of worker threads defined in WT_SMAX.

WT_IALLOCHI
The peak number of worker threads concurrently allocated to connection manager threads. This number represents a high water mark for WT_CALLOC.

Statistics for throughput analysis

Look at these key statistics when considering transaction throughput through the Gateway daemon.

GD_IALLREQ
The number of API calls (ECI, ESI, EPI) and XA requests that have been processed. Successful and failed requests are included. Administrative requests and handshakes are excluded.

GD_ILUWTXNC
The number of extended LUW-based transactions that were committed. This statistic returns information about 1–phase commit transactions.

GD_ILUWTXNR
The number of extended LUW-based transactions that were rolled back. This statistic returns information about 1–phase commit transactions.

GD_IREQDATA
The amount of request data in bytes received from client applications. All requests are included.

GD_IRESPDATA
The amount of response data in bytes sent to client applications. All responses are included.

GD_IRUNTIME
The time in seconds since the last reset event, or age of the current interval.

GD_ISYNCTXN
The number of successful SYNCONRETURN transactions.

GD_LALLREQ
The number of API calls (ECI, ESI, EPI) and XA requests that have been processed. Successful and failed requests are included. Administrative requests and handshakes are excluded.

GD_LLUWTXNC
The number of extended LUW-based transactions that were committed. This statistic returns information about 1–phase commit transactions.

GD_LLUWTXNR
The number of extended LUW-based transactions that were rolled back. This statistic returns information about 1–phase commit transactions.

GD_LREQDATA
The amount of request data in bytes received from client applications. All requests are included.

GD_LRESPDATA
The amount of response data in bytes sent to client applications. All responses are included.

GD_LRUNTIME
The length of time in seconds since the Gateway daemon successfully initialized.

GD_LSYNCTXN
The number of successful SYNCONRETURN transactions.

CICS TG plug-in for CICS Explorer

The CICS TG perspective of the CICS Explorer includes a Gateway daemons view, CICS connections view, and a CICS TG Explorer view.

To download the CICS TG plug-in see CICS Explorer.

Appendix. Data conversion

Character data might sometimes have to be converted as it is passed between a client and CICS. For example data conversion would be required if the data on the client is encoded in ASCII format but in EBCDIC format on CICS. Data conversion is performed by the CICS server.

Data conversion is controlled by ASCII and EBCDIC encoding schemes. Each encoding scheme is identified by a CCSID (Coded Character Set Identifier) that defines a set of graphic characters, and a CPGID (Code Page Global Identifier) that specifies the code points used to represent the graphic characters. For more information see *IBM Character Data Representation Architecture Reference and Registry* (SC09-2190).

Data managed by a CICS server can be accessed from client systems that use different ASCII encoding schemes. In this situation, each client system supplies a CCSID tag to CICS to ensure that the data is converted correctly.

Supported conversions

The method used to perform data conversion depends on the server platform.

The range of data conversions supported also depends on the platform. The following information is taken from *Communicating from CICS on System/390®*, and is provided as a guide; check the current copy of the book for full information. ASCII and EBCDIC CCSIDs are assigned to geographic or linguistic groups.

Data conversion is supported between ASCII and EBCDIC where both CCSIDs belong to the same group, as shown in this table.

Table 34. Data conversion support

Geographic group	Country	CCSIDs supported
Arabic	Arabic	Arabic client and server CCSIDs
Baltic Rim	Latvia, Lithuania	Baltic Rim client and server CCSIDs
Cyrillic	Eastern Europe: Bulgaria, Russia, former Yugoslavia	Cyrillic client and server CCSIDs
Estonian	Estonia	Estonian client and server CCSIDs
Greek	Greece	Greek client and server CCSIDs
Hebrew	Israel	Hebrew client and server CCSIDs
Japanese	Japan	Japanese ASCII and EBCDIC
Korean	Korea	Korean ASCII and EBCDIC
Latin-1 and Latin-9	USA, Western Europe, and many other countries.	Latin-1 and Latin-9 client and server CCSIDs
Latin-2	Eastern Europe: Albania, Czech Republic, Hungary, Poland, Romania, Slovakia, former Yugoslavia	Latin-2 ASCII and server CCSIDs
Latin-5	Turkey	Latin-5 client and server CCSIDs

Table 34. Data conversion support (continued)

Geographic group	Country	CCSIDs supported
Simplified Chinese	People's Republic of China	Simplified Chinese ASCII and EBCDIC
Traditional Chinese	Taiwan	Traditional Chinese ASCII and EBCDIC
Vietnamese	Vietnam	Vietnamese client and server CCSIDs

The tables in the links above list the CCSIDs supported for each group. For each CCSID, they show:

- The value to be specified for the CLINTCP or SRVERCP keyword.
- The code page identifier or identifiers (CPGIDs).
- The current CICS on System z products that support the CCSID. Three levels of support are defined: “Base”, “T01”, and “T02”.

  Base
  - CICS Transaction Server for z/OS
  - CICS TS for VSE

  T01
  - CICS Transaction Server for z/OS
  - CICS TS for VSE

  T02
  - CICS Transaction Server for z/OS

Arabic

A list of the Arabic client and server CCSIDs.

Table 35. Arabic client CCSIDs

CLINTCP	in	CCSID	CPGID	Comments
864	Base	00864	00864	PC data: Arabic
1089 8859-6	Base	01089	01089	ISO 8859-6: Arabic
1256	T01	01256	01256	MS Windows: Arabic
5352	T02	05352	05352	MS Windows: Arabic, version 2 with euro
17248	T02	17248	00864	PC Data: Arabic with euro

Table 36. Arabic server CCSIDs

SRVERCP	in	CCSID	CPGID	Comments
420	Base	00420	00420	Host: Arabic
16804	T02	16804	00420	Host: Arabic with euro

Note: Data conversion does not change the direction of Arabic data.

Baltic Rim

A list of the Baltic Rim client and server CCSIDs, which includes Latvia and Lithuania.

Table 37. Baltic Rim client CCSIDs

CLINTCP	in	CCSID	CPGID	Comments
901	T02	00901	00901	PC data: Latvia, Lithuania; with euro
921	T01	00921	00921	PC data: Latvia, Lithuania
1257	T01	01257	01257	MS Windows: Baltic Rim
5353	T02	05353	05353	MS Windows: Baltic Rim, version 2 with euro

Table 38. Baltic Rim server CCSIDs

SRVERCP	in	CCSID	CPGID	Comments
1112	T01	01112	01112	Host: Latvia, Lithuania
1156	T02	01156	01156	Host: Latvia, Lithuania; with euro

Cyrillic

A list of the Cyrillic client and server CCSIDs for Eastern Europe, which includes Bulgaria, Russia, and Yugoslavia.

Table 39. Cyrillic client CCSIDs

CLINTCP	in	CCSID	CPGID	Comments
808	T02	00808	00808	PC data: Cyrillic, Russia; with euro
848	T02	00848	00848	PC data: Cyrillic, Ukraine; with euro
849	T02	00849	00849	PC data: Cyrillic, Belarus; with euro
855	Base	01235	00855	PC data: Cyrillic
866	Base	00866	00866	PC data: Cyrillic, Russia
872	T02	00872	00872	PC data: Cyrillic with euro
915 8859-5	Base	00915	00915	ISO 8859-5: Cyrillic
1124	T02	01124	01124	8-bit: Cyrillic, Belarus
1125	T02	01125	01125	PC Data: Cyrillic, Ukraine
1131	T02	01131	01131	PC Data: Cyrillic, Belarus
1251	T01	01251	01251	MS Windows: Cyrillic
5347	T02	05347	05347	MS Windows: Cyrillic, version 2 with euro

Table 40. Cyrillic server CCSIDs

SRVERCP	in	CCSID	CPGID	Comments
1025	Base	01025	01025	Host: Cyrillic multilingual
1123	T02	01123	01123	Host: Cyrillic Ukraine
1154	T02	01154	01154	Host: Cyrillic multilingual; with euro
1158	T02	01158	01158	Host: Cyrillic Ukraine; wtih euro

Estonian

A list of the Estonian client and server CCSIDs.

Table 41. Estonian client CCSIDs

CLINTCP	in	CCSID	CPGID	Comments
902	T02	00902	00902	PC data: Estonia with euro
922	T01	00922	00922	PC data: Estonia
1257	T01	01257	01257	MS Windows: Baltic Rim
5353	T02	05353	05353	MS Windows: Baltic Rim, version2 with euro

Table 42. Estonian server CCSIDs

SRVERCP	in	CCSID	CPGID	Comments
1122	T01	01122	01122	Host: Estonia
1157	T01	01157	01157	Host: Estonia with euro

Greek

A list of the Greek client and server CCSIDs.

Table 43. Greek client CCSIDs

CLINTCP	in	CCSID	CPGID	Comments
813 8859-7	Base	00813	00813	ISO 8859-7: Greece
869	Base	00869	00869	PC data: Greece
1253	T01	01253	01253	MS Windows: Greece
4909	T02	04909	00813	ISO 8859-7: Greece with euro
5349	T02	05349	01253	MS Windows: Greece, version 2 with euro
9061	T02	09061	00869	PC Data: Greece with euro

Table 44. Greek server CCSIDs

SRVERCP	in	CCSID	CPGID	Comments
875	Base	00875	00875	Host: Greece
4971	T02	04971	00875	Host: Greece with euro

Hebrew

A list of the Hebrew client and server CCSIDs.

Table 45. Hebrew client CCSIDs

CLINTCP	in	CCSID	CPGID	Comments
856	Base	00856	00856	PC data: Hebrew
862	T02	00862	00862	PC data: Hebrew (upgrade)
867	T02	00867	00867	PC Data: Hebrew with euro
916 8859-8	Base	00916	00916	ISO 8859-8: Hebrew

Table 45. Hebrew client CCSIDs (continued)

CLINTCP	in	CCSID	CPGID	Comments
1255	T01	01255	01255	MS Windows: Hebrew
5351	T02	05351	05351	MS Windows: Hebrew, version 2 with euro

Table 46. Hebrew server CCSIDs

SRVERCP	in	CCSID	CPGID	Comments
424	Base	00424	00424	Host: Hebrew
803	T02	00803	00803	Host: Hebrew (Character Set A)
4899	T02	04899	00803	Host: Hebrew (Character Set A) with euro
12712	T02	12712	00424	Host: Hebrew with euro and new sheqel

Note: Data conversion does not change the direction of Hebrew data.

Japanese

A list of the Japanese ASCII and EBCDIC.

Table 47. Japanese ASCII

CLINTCP	in	CCSID	CPGID	Comments
932	Base	00932	1. 00897 2. 00301	1. PC data: SBCS 2. PC data: DBCS including 1880 user-defined characters
942	Base	00942	1. 01041 2. 00301	1. PC data: Extended SBCS 2. PC data: DBCS including 1880 user-defined characters
943	T01	00943	1. 00897 2. 00941	1. PC data: SBCS 2. PC data: DBCS for Open environment including 1880 IBM user-defined characters
954 EUCJP	Base	00954	1. 00895 2. 00952 3. 00896 4. 00953	1. G0: JIS X201 Roman 2. G1: JIS X208-1990 3. G1: JIS X201 Katakana 4. G1: JIS X212
5050	T02	05050	1. 00895 2. 00952 3. 00896 4. 00953	1. G0: JIS X201 Roman 2. G1: JIS X208-1990 3. G1: JIS X201 Katakana 4. G1: JIS X212

Table 48. Japanese EBCDIC

SRVERCP		CCSID	CPGID	Comments
930	Base	00930	1. 00290 2. 00300 3. 00290 4. 00300	1. Katakana Host: extended SBCS 2. Kanji Host: DBCS including 4370 user-defined characters 3. Katakana Host: extended SBCS 4. Kanji Host: DBCS including 1880 user-defined characters
931	Base	00931	1. 00037 2. 00300	1. Latin Host: SBCS 2. Kanji Host: DBCS including 4370 user-defined characters

Table 48. Japanese EBCDIC (continued)

SRVERCP		CCSID	CPGID	Comments
939	Base	00939	1. 01027 2. 00300 3. 01027 4. 00300	1. Latin Host: extended SBCS 2. Kanji Host: DBCS including 4370 user-defined characters 3. Latin Host: extended SBCS 4. Kanji Host: DBCS including 1880 user-defined characters
1390	T02	01390	1. 00290 2. 00300	1. Katakana Host: extended SBCS; with euro 2. Kanji Host: DBCS including 6205 user-defined characters
1399	T02	01399	1. 01027 2. 00300	1. Latin Host: extended SBCS; with euro 2. Kanji Host: DBCS including 4370 user-defined characters; with euro

Korean

A list of the Korean ASCII and EBCDIC.

Table 49. Korean ASCII

CLINTCP	in	CCSID	CPGID	Comments
934	Base	00934	1. 00891 2. 00926	1. PC data: SBCS 2. PC data: DBCS including 1880 user-defined characters
944	Base	00944	1. 01040 2. 00926	1. PC data: Extended SBCS 2. PC data: DBCS including 1880 user-defined characters
949	Base	00949	1. 01088 2. 00951	1. IBM KS Code - PC data: SBCS 2. IBM KS code - PC data: DBCS including 1880 user-defined characters
970 EUCKR	Base	00970	1. 00367 2. 00971	1. G0: ASCII 2. G1: KSC X5601-1989 including 1880 user-defined characters
1363	T01	01363	1. 01126 2. 01362	1. PC data: MS Windows Korean SBCS 2. PC data: MS Windows Koran DBCS including 11172 full Hangul

Table 50. Korean EBCDIC

SRVERCP	in	CCSID	CPGID	Comments
933	Base	00933	1. 00833 2. 00834	1. Host: Extended SBCS 2. Host: DBCS including 1880 user-defined characters and 11172 full Hangul characters
1364	T02	01364	1. 00833 2. 00834	1. Host: Extended SBCS 2. Host: DBCS including 1880 user-defined characters and 11172 full Hangul characters

Latin-1 and Latin-9

A list of the Latin-1 and Latin-9 client and server CCSIDs, which includes the United States of America, Western Europe, and many other countries.

Table 51. Latin-1 client CCSIDs

CLINTCP	in	CCSID	CPGID	Comments
437	Base	00437	00437	PC data: PC Base; USA, many other countries
819 8859-1	Base	00819	00819	ISO 8859-1: Latin-1 countries
850	Base	00850	00850	PC data: Latin-1 countries
858	T01	00858	00858	PC data: Latin-1 countries; with euro
923	T01	00923	00923	ISO 8859-15: Latin-9
924	T02	00924	00924	ISO 8859-15: Latin-9
1047	T02	01047	01047	Host: Open Edition Latin-1
1252	T01	01252	01252	MS Windows: Latin-1 countries
5348	T01	05348	01252	MS Windows: Latin-1 countries, version 2 with euro

Table 52. Latin-1 and Latin-9 server CCSIDs

SRVERCP	in	CCSID	CPGID	Comments
037	Base	00037	00037	Host: USA, Canada (ESA), Netherlands, Portugal, Brazil, Australia, New Zealand
273	Base	00273	00273	Host: Austria, Germany
277	Base	00277	00277	Host: Denmark, Norway
278	Base	00278	00278	Host: Finland, Sweden
280	Base	00280	00280	Host: Italy
284	Base	00284	00284	Host: Spain, Latin America (Spanish)
285	Base	00285	00285	Host: United Kingdom
297	Base	00297	00297	Host: France
500	Base	00500	00500	Host: Belgium, Canada (AS/400®), Switzerland, International Latin-1
871	Base	00871	00871	Host: Iceland
924	T01	00924	00924	Host: Latin-9
1047	T01	01047	01047	Host: Open Edition Latin-1
1140	T01	01140	01140	Host: USA, Canada (ESA), Netherlands, Portugal, Brazil, Australia, New Zealand; with euro
1141	T01	01141	01141	Host: Austria, Germany; with euro
1142	T01	01142	01142	Host: Denmark, Norway; with euro
1143	T01	01143	01143	Host: Finland, Sweden; with euro
1144	T01	01144	01144	Host: Italy; with euro
1145	T01	01145	01145	Host: Spain, Latin America (Spanish); with euro
1146	T01	01146	01146	Host: United Kingdom; with euro
1147	T01	01147	01147	Host: France; with euro
1148	T01	01148	01148	Host: Belgium, Canada (AS/400), Switzerland, International Latin-1; with euro
1149	T01	01149	01149	Host: Iceland; with euro

Note: Conversions are supported between non euro-supported CCSIDs and euro-supported CCSIDs. These should be used with care because:

- The international currency symbol in each non euro-supported EBCDIC CCSID (for example, 00500) has been replaced by the euro symbol in the equivalent euro-supported EBCDIC CCSID (for example, 01148).
- The dotless *i* in non euro-supported ASCII CCSID 00850 has been replaced by the euro symbol in the equivalent euro-supported ASCII CCSID 00858.

Latin-2

A list the Latin-2 ASCII and server CCSIDs for Eastern Europe, which includes Albania, Czech Republic, Hungary, Poland, Romania, Slovakia, Yugoslavia, and former Yugoslavia.

Table 53. Latin-2 ASCII

CLINTCP	in	CCSID	CPGID	Comments
852	Base	00852	00852	PC data: Latin-2 multilingual
912 8859-2	Base	00912	00912	ISO 8859-2: Latin-2 multilingual
1250	T01	01250	01250	MS Windows: Latin-2
5346	T02	05346	01250	MS Windows: Latin-2, version 2 with euro
9044	T02	09044	00852	PC data: Latin-2 multilingual with euro

Table 54. Latin-2 Server CCSIDs

SRVERCP	in	CCSID	CPGID	Comments
500	T02	00500	00500	Host: International Latin-1
870	Base	00870	00870	Host: Latin-2 multilingual
1148	T02	01148	01148	Host: International Latin-1 with euro
1153	T02	01153	01153	Host: Latin-2 multilingual with euro

Note: Conversions are supported for some combinations of Latin-2 ASCII CCSIDs and Latin-1 EBCDIC CCSIDs.

Latin-5

A list of the Latin-5 client and server CCSIDs, which includes Turkey.

Table 55. Latin-5 client CCSIDs

CLINTCP	in	CCSID	CPGID	Comments
857	Base	00857	00857	PC data: Latin-5 (Turkey)
920 8859-9	Base	00920	00920	ISO 8859-9: Latin-5 (ECMA-128, Turkey TS-5881)
1254	T01	01254	01254	MS Windows: Turkey
5350	T02	05350	01254	MS Windows: Turkey, version 2 with euro
9049	T02	09049	00857	PC data: Latin-5 (Turkey) with euro

Table 56. Latin-5 server CCSIDs

SRVERCP	in	CCSID	CPGID	Comments
1026	Base	01026	01026	Host: Latin-5 (Turkey)
1155	T02	01155	01155	Host: Latin-5 (Turkey) with euro

Simplified Chinese

A list of the simplified Chinese ASCII and EBCDIC.

Table 57. Simplified Chinese ASCII

CLINTCP	in	CCSID	CPGID	Comments
946	Base	00946	1. 01042 2. 00928	1. PC data: Extended SBCS 2. PC data: DBCS including 1880 user-defined characters
1381	Base	01381	1. 01115 2. 01380	1. PC data: Extended SBCS (IBM GB) 2. PC data: DBCS (IBM GB) including 31 IBM-selected, 1880 user-defined characters
1383 EUCCN	T01	01383	1. 00367 2. 01382	1. G0: ASCII 2. G1: GB 2312-80 set
1386	T01	01386	1. 01114 2. 01385	1. PC data: S-Chinese GBK and T-Chinese IBM BIG-5 2. PC data: S-Chinese GBK

Table 58. Simplified Chinese EBCDIC

SRVERCP	in	CCSID	CPGID	Comments
935	Base	00935	1. 00836 2. 00837	1. Host: Extended SBCS 2. Host: DBCS including 1880 user-defined characters
1388	T02	01388	1. 00836 2. 00837	1. Host: Extended SBCS 2. Host: DBCS including 1880 user-defined characters
9127	T02	09127	1. 00836 2. 00837	1. Host: Extended SBCS 2. Host: DBCS including 1880 user-defined characters

Traditional Chinese

A list of the traditional Chinese ASCII and EBCDIC.

Table 59. Traditional Chinese ASCII

CLINTCP	in	CCSID	CPGID	Comments
938	Base	00938	1. 00904 2. 00927	1. PC data: SBCS 2. PC data: DBCS including 6204 user-defined characters
948	Base	00948	1. 01043 2. 00927	1. PC data: Extended SBCS 2. PC data: DBCS including 6204 user-defined characters
950 BIG5	Base	00950	1. 01114 2. 00947	1. PC data: SBCS (IBM BIG5) 2. PC data: DBCS including 13493 CNS, 566 IBM selected, 6204 user-defined characters
964 EUCTW	Base	00964	1. 00367 2. 00960 3. 00961	1. G0: ASCII 2. G1: CNS 11643 plane 1 3. G1: CNS 11643 plane 2

Table 59. Traditional Chinese ASCII (continued)

CLINTCP	in	CCSID	CPGID	Comments
1370	T02	01370	1. 01114 2. 00947	1. PC data: Extended SBCS; with euro 2. PC data: DBCS including 6204 user-defined characters; with euro

Table 60. Traditional Chinese EBCDIC

SRVERCP	in	CCSID	CPGID	Comments
937	Base	00937	1. 00037 2. 00835	1. Host: Extended SBCS 2. Host: DBCS including 6204 user-defined characters
1371	T02	01371	1. 01159 2. 00835	1. Host: Extended SBCS; with euro 2. Host: DBCS including 6204 user-defined characters; with euro

Vietnamese

A list of the Vietnamese client and server CCSIDs.

Table 61. Vietnamese client CCSIDs

CLINTCP	in	CCSID	CPGID	Comments
1129	T02	01129	01129	ISO-8: Vietnamese
1163	T02	01163	01163	ISO-8: Vietnamese with euro
1258	T02	01258	01258	MS Windows: Vietnamese
5354	T02	05354	01258	MS Windows: Vietnamese, version 2 with euro

Table 62. Vietnamese server CCSIDs

SRVERCP	in	CCSID	CPGID	Comments
1130	T02	01130	01130	Host: Vietnamese
1164	T02	01164	01164	Host: Vietnamese with euro

Unicode data

CICS on System z provides limited support for Unicode-encoded character data. This support allows workstations to share UCS-2 or UTF-8 encoded data with the operating system provided that no conversion is required.

Table 63. Unicode

CLINTCP SRVERCP	in	CCSID	CPGID	Comments
1200 UCS-2	T01	01200	01400	UCS-2 level 3, maximal (growing) character set
1208 UTF-8	T01	01200	01400	UTF-8 based on UCS-2 level 3, maximal (growing) character set
13488	T01	13488	01400	UCS-2 level 1 (level 3 tolerant), subset (fixed) character set

Product library and related literature

The CICS Transaction Gateway product library contains information on administration, messages and programming; this information is available in this information center, and is also available in PDF form. IBM Redbooks publications provide a further source of information about working with CICS Transaction Gateway.

CICS Transaction Gateway books

The books in the library cover administration, programming and messages.

- *CICS Transaction Gateway: Windows Administration, SC34-7055-00* describes the administration of the CICS Transaction Gateway for Windows.
- *CICS Transaction Gateway: UNIX and Linux Administration, SC34-7054-00* describes the administration of the CICS Transaction Gateway for UNIX and Linux.
- *CICS Transaction Gateway for Multiplatforms: Programming Guide*, SC34-7056-00 introduces programming for the CICS Transaction Gateway and provides information on working with user applications in a client/server environment.
- *CICS Transaction Gateway for Multiplatforms: Programming Reference*, SC34-7057-00 provides information on the APIs for the programming languages supported by the CICS Transaction Gateway for UNIX, Linux and Windows.

  Additional HTML pages contain JAVA programming reference information.
- *CICS Transaction Gateway: Messages*, SC34-7061-00 describes the error messages that can be generated by the CICS Transaction Gateway.

Sample configuration documents

Several sample configuration documents are available as PDFs. These documents give step-by-step guidance for configuring CICS Transaction Gateway for communication with CICS servers, using various protocols. They provide detailed instructions that extend the information in the CICS Transaction Gateway library.

- *Migrating TCP62 connections to Communications Server Remote API Client* , GC34-6889
- *Configuring Enterprise Extender Connections*, GC34-6976

Sample configuration documents

Several sample configuration documents are available in portable document format (PDF).

These documents give step-by-step guidance for configuring CICS Transaction Gateway for communication with CICS servers, using various protocols. They provide detailed instructions that extend the information in the CICS Transaction Gateway library.

Visit the following Web site:

`www.ibm.com/software/cics/ctg`

and follow the **Library** link.

IBM Redbooks publications

IBM Redbook titles are available on a wide range of subjects relevant to CICS Transaction Gateway programming, installation, operation and troubleshooting.

The following International Technical Support Organization (ITSO) Redbook publication contains many examples of client/server configurations:

- *CICS Transaction Gateway V5 - The WebSphere Connector for CICS, SG24-6133* describes how to use the different protocols (TCP/IP, TCP62, APPC and EXCI) for communication with CICS, and how to securely connect a Java client application to a CICS region.
- *Revealed! Architecting Web Access to CICS, SG24-5466* is intended for IT architects who select, plan, and design SOA solutions that make use of CICS assets
- *Enterprise JavaBeans for z/OS and z/OS CICS Transaction Server V2.2, SG24-6284* describes the EJB and the way it has been implemented within the CICS architecture, also describes how to set up and configure a CICS region to support EJBs
- *Java Connectors for CICS: Featuring the J2EE Connector Architecture, SG24-6401* provides information on developing J2EE applications.
- *Systems Programmer's Guide to Resource Recovery Services (RRS), SG24-6980-00* describes how to use RRS in various scenarios.
- *Communications Server for z/OS V1R2 TCP/IP Implementation Guide, SG24-6517-00* provides information on using Communications Server for z/OS V1R2, including load balancing.
- *Redpaper: Transactions in J2EE, REDP-3659-00* discusses transactions in the J2EE environment, including one-phase commit and two-phase commit XA transactions.
- *Exploring Systems Monitoring for CICS Transaction Gateway V7.1 for z/OS, SG24-7562-00* looks at product installation and customization, and also covers systems monitoring for CICS Transaction Gateway using IBM Tivoli OMEGAMON XE, and statistics provided by CICS Performance Analyzer.
- *CICS Transaction Gateway for z/OS Version 6.1 SG24-7161-00* introduces the new facilities of the CICS TG for z/OS V6.0 and V6.1, which provide improvements in the areas of transactional integration, systems management, performance, security, and ease of use.

The ITSO Redbooks are available from various sources. For the latest information, see:

`www.ibm.com/redbooks/`

Other useful information

Other sources of useful information include the CICS Transaction Server information center and associated publications.

The CICS Transaction Server for z/OS V4.1 information center is located at:

http://publib.boulder.ibm.com/infocenter/cicsts/v4r1/index.jsp

CICS Transaction Server publications

The CICS Transaction Server books on security, inter-product communication and problem determination also provide a useful source of information.

CICS Transaction Server for z/OS RACF Security Guide, SC34-7003

CICS inter-product communication

The following books describe the intercommunication facilities of the CICS server products:

- *CICS Family: Interproduct Communication*, SC34–6853
- *CICS Transaction Server for z/OS CICS External Interfaces Guide*, SC34-7019
- *CICS Transaction Server for z/OS: Intercommunication Guide*, SC34-7018
- *CICS TS for VSE: Intercommunication Guide*, SC33-0701
- *CICS Transaction Server for iSeries: Intercommunication*, SC41-5456
- *TXSeries for Multiplatforms: Intercommunication Guide*, SC34-6644

The first book above is a CICS family book containing a platform-independent overview of CICS inter-product communication.

CICS problem determination

The following books describe the problem determination facilities of the CICS server products:

- *Transaction Server for Windows Problem Determination*, GC34-6210
- *CICS Transaction Server for z/OS Problem Determination Guide*, SC34-7034
- *CICS TS for VSE 2.3 Problem Determination Guide*, SC33-0716
- *CICS Transaction Server for iSeries: Problem Determination*, SC41-5453
- *TXSeries for Multiplatforms: Problem Determination Guide*, SC34-6636

APPC-related publications

Publications related to APPC also provide a useful source of additional information.

If you are using HP-UX or Solaris, see the SNA documentation for your platform.

IBM products

The IBM Communications Server and IBM Personal Communications library pages provide an additional source of information related to APPC.

IBM Communications Server

See this Web page:

`www.ibm.com/software/network/commserver/library`

IBM Personal Communications

See this Web page:

`www.ibm.com/software/network/pcomm/library`

Systems Network Architecture (SNA)

The IBM SNA publications also provide a useful source of information related to APPC.

The IBM SNA publications are:

- *SNA Formats*, GA27-3136

- *Systems Network Architecture Technical Overview*, GC30-3073
- *Guide to SNA over TCP/IP*, SC31-6527

Accessibility

Accessibility features help users with a physical disability, for example restricted mobility or limited vision, to use information technology products successfully. CICS Transaction Gateway provides accessibility by enabling keyboard-only operation.

For more information about the IBM commitment to accessibility, visit the IBM Accessibility Center.

Installation

The InstallAnywhere wizard is not fully accessible to screen readers.

To use the installer with a screen reader you must use console mode installation. Console mode installation is run at a command prompt and is specified by using the -i console option.

The console mode option displays text over a number of screens and you can make various choices during the installation process. The command prompt interface does not have a cursor to navigate over the displayed text. When you use the JAWS screen reader the displayed text can be repeated with the command to read the current window using the key sequence Insert+B.

The first screen displayed by the installer is to select language, the default varies with the system's regional settings. To avoid the language selection screen use the -l <lang> option; where land can be one of the following:

- `de` to specify German
- `en` to specify English
- `es` to specify Spanish
- `fr` to specify French
- `it` to specify Italian
- `ja` to specify Japanese
- `ko` to specify Korean
- `tr` to specify Turkish
- `zh_CN` to specify Chinese

For example; to install with the console interface in French, enter the command:

```
installer -i console -l fr
```

Configuration Tool accessibility

The configuration file uses the number sign (#) character to denote a comment; consider configuring your screen reader accordingly.

The recommended way to configure the CICS Transaction Gateway if you use a screen reader is by editing the configuration file.

Components

Each component in the Configuration Tool has a name and description for screen readers.

Keys

Keyboard shortcuts provide an alternative way of working with the Configuration Tool.

Alt, Space
Press and release the **Alt** key, then press **Space**, to open the window menu. This allows you to move, size, maximize or minimize the window.

Ctrl+key
Certain actions have shortcuts assigned to them. Hold down the Ctrl key and type the letter to do the action.

Action	Ctrl+key
Activate a link.	Spacebar
Copy the selected value	C except in the following locales: **Locale Key** **German** K **Turkish** P
Create a new configuration	N
Create a new server definition	W except in the following locales: **Locale Key** **German** N **Italian** V **Spanish** V **Turkish** S
Cut the selected value	U except in the following locales: **Locale Key** **Italian** G **Spanish** O **Turkish** E
Cycle between the navigation panel, objects on the settings panel, and the buttons	Use this key combination on panels that contain a table. Otherwise use **Tab**.
Next link or other focusable object	T

Action	Ctrl+key
Open an existing configuration	O except in the following locales: **Locale Key** **German** F **Italian** A **Spanish** B **Turkish** A
Paste	P except in the following locales: **Locale Key** **French** L **German** F **Italian** I **Turkish** P
Previous link or other focusable object	Shift+T
Save the current configuration	S except in the following locales: **Locale Key** **Spanish** G **Italian** L
Scroll left	Page Up
Scroll right	Page Down

Arrows (left and right)

1. (Applies if the menus are active.) Move to a different menu.
2. (Applies if the buttons are active.) Cycle through the buttons.
3. (Applies if the navigation panel is active.) If a node contains subnodes, the left arrow collapses the node, the right arrow expands it. If a node cannot be expanded further, pressing the right arrow moves down to the next node. If a subnode is selected, pressing the left arrow moves to the parent node.

Arrows (up and down)

1. (Applies if the navigation panel is active.) Move up and down through the navigation panel.
2. (Applies if the buttons are active.) Cycle through the buttons.
3. (Applies if the Authorized Hosts list box is active.) Move through the entries in the list. If only one entry is in the list, press the **Home** key to select it.

Escape
Closes the menu.

F2 Select and change an editable number field in a table.

F6 (Applies if the navigation panel or the settings panel is active.) Toggles between the navigation panel and the settings panel.

F8 (Applies if the navigation panel or the settings panel is active.) Allows you to gain control of the split between the navigation panel and settings panel areas. After pressing F8, use the arrow keys to move the split:

- Press the left arrow key to move the split left (decrease the width of the navigation panel, increase the width of the settings panel).
- Press the right arrow key to move the split right (increase the width of the navigation panel, decrease the width of the settings panel).

F10
Opens the file menu.

Home
Selects the first entry in the Authorized Hosts list box.

Page Up
Scroll up.

Page Down
Scroll down.

Scroll bars
You cannot use the keyboard to control scroll bars in the navigation panel. Use the up and down arrow keys to navigate the tree; settings for the selected item are shown in the settings panel.

Shift+Tab
If the cursor is on a field in the settings panel other than the first field, moves backwards through the fields. Otherwise, toggles between the navigation panel and the settings panel.

Space

1. Activates a button.
2. Selects a check box.
3. Selects a node in the tree.

Tab
Cycles through the tree, fields in the settings panel, and the buttons.

Customizing colors and fonts

Use the **Settings** option on the menu to change colors and fonts.

Fields that contain errors are displayed by default in the font warning color, preceded by an asterisk. Use the **Settings->Font** option on the menu to change the warning color.

Starting the Gateway daemon

You can start the Gateway daemon from a command prompt using a screen reader.

In some Telnet sessions, the screen reader might reread CICS Transaction Gateway log output or the command prompt after the CICS Transaction Gateway has started. This behavior is expected, and does not mean that the CICS Transaction Gateway has failed to start.

To determine if the CICS Transaction Gateway started correctly, check for the message:

```
'CTG6512I CICS Transaction Gateway initialization complete'.
```

If the CICS Transaction Gateway did not start successfully, this message is produced:

```
'CTG6513E CICS Transaction Gateway failed to initialize'.
```

cicsterm

Although cicsterm is accessible, it relies on the application that is being processed to define an accessible 3270 screen.

Keyboard mapping depends on the terminal type that you are using.

The bottom row of cicsterm contains status information. The following list shows this information, as it appears from left to right:

Status For example, **1B** is displayed while cicsterm is connecting to a server. Displayed at columns 1 – 3.

Terminal name
Also referred to as *LU Name*. Columns 4 – 7.

Action
For example, **X-System**, indicating that you cannot enter text in the terminal window because cicsterm is waiting for a response from the server. Columns 9 – 16.

Error number
Errors in the form CCLNNNN, relating to the CICS Transaction Gateway. Columns 17 – 24.

Server name
The server to which cicsterm is connected. Columns 27 – 35.

Uppercase
An up arrow is displayed when the Shift key is pressed. Column 42.

Caps Lock
A capital A is displayed when Caps Lock is on. Column 43.

Insert on
The caret symbol (^) is displayed if text will be inserted, rather than overwriting existing text. If you have difficulty seeing the caret, change the font face and size, or use a screen magnifier to increase the size of the status line. Column 52.

Cursor position
The cursor position, in the form ROW/COLUMN, where ROW is a two-digit number, and COLUMN a three-digit number. The top left of the screen is 01/001. Column 75–80.

Note: You might need to change the default behavior of your screen reader if it reads only the last digit of the cursor position. Customize your screen reader to specify that columns 75–80 of the status row are to be treated as one field. This will cause the full area to be read when any digit changes.

Glossary

This glossary defines the terms and abbreviations used in CICS Transaction Gateway and in the information centers.

A

abnormal end of task (abend)
The termination of a task, job, or subsystem because of an error condition that recovery facilities cannot resolve.

Advanced program-to-program communication (APPC)
An implementation of the SNA/SDLC LU 6.2 protocol that allows interconnected systems to communicate and share the processing of programs. The Client daemon uses APPC to communicate with CICS systems.

APAR See *Authorized program analysis report*.

API See *application programming interface*.

APPC See *Advanced program-to-program communication*.

application programming interface (API)
A functional interface that allows an application program that is written in a high-level language to use specific data or functions of the operating system or another program.

APPLID

1. On CICS Transaction Gateway: The application identifier that is used to identify connections on the CICS server and tasks in a CICSplex. See also *APPLID qualifier* and *fully-qualified APPLID*.
2. On CICS Transaction Server: The name by which a CICS system is known in a network of interconnected CICS systems. CICS Transaction Gateway application identifiers do not need to be defined in SYS1.VTAMLST. The CICS APPLID is specified in the APPLID system initialization parameter.

APPLID qualifier
Optionally used as a high-level qualifier for the APPLID to form a fully-qualified APPLID. See also *APPLID* and *fully-qualified APPLID*.

ARM See *automatic restart manager*.

Authorized program analysis report (APAR)
A request for correction of a defect in a current release of an IBM-supplied program.

ATI See *automatic transaction initiation*.

attach In SNA, the request unit that flows on a session to initiate a conversation.

Attach Manager
The component of APPC that matches attaches received from remote computers to accepts issued by local programs.

autoinstall
A method of creating and installing resources dynamically as terminals log on, and deleting them at logoff.

automatic restart manager (ARM)
A z/OS recovery function that can improve the availability of specific batch jobs or started tasks, and therefore result in faster resumption of productive work.

automatic transaction initiation (ATI)
The initiation of a CICS transaction by an internally generated request, for example, the issue of an EXEC CICS START command or the reaching of a transient data trigger level. CICS resource definition can associate a trigger level and a transaction with a transient data destination. When the number of records written to the destination reaches the trigger level, the specified transaction is automatically initiated.

B

bean A definition or instance of a JavaBeans component. See also *JavaBeans*.

bean-managed transaction
A transaction where the JEE bean itself is responsible for administering transaction tasks such as committal or rollback. See also *container-managed transaction*.

BIND command
In SNA, a request to activate a session between two logical units (LUs).

business logic
The part of a distributed application that is concerned with the application logic rather than the user interface of the application. Compare with *presentation logic*.

C

CA See *certificate authority*.

CCIN The CCIN transaction is invoked by the Client daemon, for each TCP/IP or SNA connection established. CCIN installs a Client connection on the CICS server.

CCSID
Coded Character Set Identifier. A 16-bit number that includes a specific set of encoding scheme identifiers, character set identifiers, code page identifiers, and other information that uniquely identifies the coded graphic-character representation.

CTIN The CTIN transaction is invoked by the Client daemon to install a Client terminal definition on the CICS server.

callback
A way for one thread to notify another application thread that an event has happened.

certificate authority (CA)
In computer security, an organization that issues certificates. The certificate authority authenticates the certificate owner's identity and the services that the owner is authorized to use. It issues new certificates and revokes certificates from users who are no longer authorized to use them.

change-number-of-sessions (CNOS)
An internal transaction program that regulates the number of parallel sessions between the partner LUs with specific characteristics.

channel
A channel is a set of containers, grouped together to pass data to CICS. There is no limit to the number of containers that can be added to a channel, and the size of individual containers is limited only by the amount of storage that you have available.

CICS connectivity components
A generic reference to the Client daemon, EXCI, and the IPIC protocol.

CICS connectivity components
The Client daemon, the EXCI (External CICS Interface), and the IPIC (IP Interconnectivity) protocol are collectively called the 'CICS connectivity components'. The Client daemon handles the TCP/IP and the SNA protocols.

CICS Request Exit
An exit that is invoked by the CICS Transaction Gateway for z/OS at run time to determine which CICS server to use.

CICS server name
A defined server known to CICS Transaction Gateway.

CICS TS
Abbreviation of CICS Transaction Server.

class In object-oriented programming, a model or template that can be instantiated to create objects with a common definition and therefore, common properties, operations, and behavior. An object is an instance of a class.

CLASSPATH
In the execution environment, an environment variable keyword that specifies the directories in which to look for class and resource files.

Client API
The Client API is the interface used by Client applications to interact with CICS using the Client daemon. See External Call Interface, External Presentation Interface, and External Security Interface.

Client application
The client application is a user application written in a supported programming language that uses one or more of the CICS Transaction Gateways APIs.

Client daemon
The Client daemon manages TCP/IP and SNA connections to CICS servers on UNIX, Linux, and Windows. It processes ECI, EPI, and ESI requests, sending and receiving the appropriate flows to and from the CICS server to satisfy Client application requests. It can support concurrent requests to one or more CICS servers. The CICS Transaction Gateway initialization file defines the operation of the Client daemon and the servers and protocols used for communication.

client/server
Pertaining to the model of interaction in distributed data processing in which a program on one computer sends a request to a program on another computer and awaits a response. The requesting program is called a client; the answering program is called a server.

CNOS See *Change-Number-of-Sessions*.

code page
An assignment of hexadecimal identifiers (code points) to graphic characters. Within a given code page, a code point can have only one meaning.

color mapping file
A file that is used to customize the 3270 screen color attributes on client workstations.

COMMAREA
See *communication area*.

commit phase
The second phase in a XA process. If all participants acknowledge that they are prepared to commit , the transaction manager issues the commit request. If any participant is not prepared to commit the transaction manager issues a back-out request to all participants.

communication area (COMMAREA)
A communication area that is used for passing data both between programs within a transaction and between transactions.

Configuration file
A file that specifies the characteristics of a program, system device, server or network.

connection
In data communication, an association established between functional units for conveying information.

In Open Systems Interconnection architecture, an association established by a given layer between two or more entities of the next higher layer for the purpose of data transfer.

In TCP/IP, the path between two protocol application that provides reliable data stream delivery service.

In Internet, a connection extends from a TCP application on one system to a TCP application on another system.

container
A container is a named block of data designed for passing information between programs. A container is a "named COMMAREA" that is not limited to 32KB. Containers are grouped together in sets called channels.

container-managed transaction
A transaction where the EJB container is responsible for administration of tasks such as committal or rollback. See also *bean-managed transaction*.

control table
In CICS, a storage area used to describe or define the configuration or operation of the system.

conversation
A connection between two programs over a session that allows them to communicate with each other while processing a transaction.

conversation security
In APPC, a process that allows validation of a user ID or group ID and password before establishing a connection.

D

daemon
A program that runs unattended to perform continuous or periodic systemwide functions, such as network control. A daemon can be launched automatically, such as when the operating system is started, or manually.

data link control (DLC)
A set of rules used by nodes on a data link (such as an SDLC link or a token ring) to accomplish an orderly exchange of information.

DBCS See *double-byte character set*.

default CICS server
The CICS server that is used if a server name is not specified on an ECI, EPI, or ESI request. The default CICS server name is defined as a product wide setting in the configuration file (ctg.ini).

dependent logical unit
A logical unit that requires assistance from a system services control point (SSCP) to instantiate an LU-to-LU session.

deprecated
Pertaining to an entity, such as a programming element or feature, that is supported but no longer recommended, and that might become obsolete.

digital certificate
An electronic document used to identify an individual, server, company, or some other entity, and to associate a public key with the entity. A digital certificate is issued by a certificate authority and is digitally signed by that authority.

digital signature
Information that is encrypted with an entity's private key and is appended to a message to assure the recipient of the authenticity and integrity of the message. The digital signature proves that the message was signed by the entity that owns, or has access to, the private key or shared secret symmetric key.

distinguished name
The name that uniquely identifies an entry in a directory. A distinguished name is made up of attribute:value pairs, separated by commas. The format of a distinguished name is defined by RFC4514. For more information, see `http://www.ietf.org/rfc/rfc4514.txt`. See also *realm name* and *identity propagation*.

distributed application
An application for which the component application programs are distributed between two or more interconnected processors.

distributed identity
User identity information that originates from a remote system. The distributed identity is created in one system and is passed to one or more other systems over a network. See also *distinguished name* and *realm name*.

distributed processing
The processing of different parts of the same application in different systems, on one or more processors.

distributed program link (DPL)
A link that enables an application program running on one CICS system to link to another application program running in another CICS system.

DLC See *data link control*.

DLL See *dynamic link library*.

domain
In the Internet, a part of a naming hierarchy in which the domain name consists of a sequence of names (labels) separated by periods (dots).

domain name
In TCP/IP, a name of a host system in a network.

domain name server
In TCP/IP, a server program that supplies name-to-address translation by mapping domain names to IP addresses. Synonymous with name server.

dotted decimal notation
The syntactical representation for a 32-bit integer that consists of four 8-bit numbers written in base 10 with periods (dots) separating them. It is used to represent IP addresses.

double-byte character set (DBCS)
A set of characters in which each character is represented by 2 bytes. Languages such as Japanese, Chinese and Korean, which contain more symbols than can be represented by 256 code points, require double-byte character sets. Because each character requires 2 bytes, the typing, display, and printing of DBCS characters requires hardware and programs that support DBCS. Contrast with *single-byte character set.*

DPL See *distributed program link*.

dynamic link library (DLL)
A collection of runtime routines made available to applications as required.

dynamic server selection (DSS)
The mapping of a logical CICS server name to an actual CICS server name at run time.

E

EBCDIC
See *extended binary-coded decimal interchange code*.

ECI See *external call interface*.

EJB See *Enterprise JavaBeans*.

emulation program
A program that allows a host system to communicate with a workstation in the same way as it would with the emulated terminal.

emulator
A program that causes a computer to act as a workstation attached to another system.

encryption
The process of transforming data into an unintelligible form in such a way that the original data can be obtained only by using a decryption process.

enterprise bean
A Java component that can be combined with other resources to create JEE applications. There are three types of enterprise beans: entity beans, session beans, and message-driven beans.

Enterprise Information System (EIS)
The applications that comprise an enterprise's existing system for handling

company-wide information. An enterprise information system offers a well-defined set of services that are exposed as local or remote interfaces or both.

Enterprise JavaBeans (EJB)
A component architecture defined by Sun Microsystems for the development and deployment of object-oriented, distributed, enterprise-level applications (JEE).

environment variable
A variable that specifies the operating environment for a process. For example, environment variables can describe the home directory, the command search path, the terminal in use, and the current time zone.

EPI See *external presentation interface*.

ESI See *external security interface*.

Ethernet
A local area network that allows multiple stations to access the transmission medium at will without prior coordination, avoids contention by using carrier sense and deference, and resolves contention by using collision detection and transmission. Ethernet uses carrier sense multiple access with collision detection (CSMA/CD).

EXCI See *external CICS interface*.

extended binary-coded decimal interchange code (EBCDIC)
A coded character set of 256 8-bit characters developed for the representation of textual data.

extended logical unit of work (extended LUW)
A logical unit of work that is extended across successive ECI requests to the same CICS server.

external call interface (ECI)
A facility that allows a non CICS program to run a CICS program. Data is exchanged in a COMMAREA or a channel as for usual CICS interprogram communication.

external communications interface (EXCI)
An MVS application programming interface provided by CICS Transaction Server for z/OS that enables a non-CICS program to call a CICS program and to pass and receive data using a COMMAREA. The CICS application program is started as if linked-to by another CICS application program.

external presentation interface (EPI)
A facility that allows a non CICS program to appear to CICS as one or more standard 3270 terminals. 3270 data can be presented to the user by emulating a 3270 terminal or by using a graphical user interface.

external security interface (ESI)
A facility that enables client applications to verify and change passwords for user IDs on CICS servers.

External Security Manager (ESM)
A security manager that operates outside CICS. For example, RACF can be used as an external security manager with CICS Transaction Server.

F

firewall
A configuration of software that prevents unauthorized traffic between a trusted network and an untrusted network.

FMH See *function management header*.

fully-qualified APPLID
Used to identify CICS Transaction Gateway connections on the CICS server and tasks in a CICSplex. It is composed of an APPLID with an optional network qualifier. See also *APPLID* and *APPLID qualifier*.

function management header (FMH)
One or more headers, optionally present in the leading request units (RUs) of an RU chain, that allow one LU to (a) select a transaction program or device at the session partner and control the way in which the end-user data it sends is handled at the destination, (b) change the destination or the characteristics of the data during the session, and (c) transmit between session partners status or user information about the destination (for example, a program or device). Function management headers can be used with LU type 1, 4, and 6.2 protocols.

G

gateway
A device or program used to connect two systems or networks.

gateway classes
The gateway classes provide APIs for ECI, EPI, and ESI that allow communication between Java client applications and the Gateway daemon.

Gateway daemon
A long-running Java process that listens for network requests from remote Client applications. It issues these requests to CICS servers using the CICS connectivity components. The Gateway daemon on z/OS processes ECI requests and on UNIX, Windows, and Linux platforms it process EPI and ESI requests as well. The Gateway daemon uses the GATEWAY section of ctg.ini for its configuration.

Gateway group
A set of Gateway daemons that share an APPLID qualifier, and where each Gateway daemon has a unique APPLID within the Gateway group.

gateway token
A token that represents a specific Gateway daemon, when a connection is established successfully. Gateway tokens are used in the C language statistics and ECI V2 APIs.

global transaction
A recoverable unit of work performed by one or more resource managers in a distributed transaction processing environment and coordinated by an external transaction manager.

H

HA group
See *highly available gateway group*.

highly available gateway group (HA group)
A Gateway group that utilizes TCP/IP load balancing, and can be viewed

as a single logical Gateway daemon. A Gateway daemon instance in a HA group can recover indoubt XA transactions on behalf of another Gateway daemon within the HA group

host A computer that is connected to a network (such as the Internet or an SNA network) and provides an access point to that network. The host can be any system; it does not have to be a mainframe.

host address
An IP address that is used to identify a host on a network.

host ID
In TCP/IP, that part of the IP address that defines the host on the network. The length of the host ID depends on the type of network or network class (A, B, or C).

host name
In the Internet suite of protocols, the name given to a computer. Sometimes, host name is used to mean the fully qualified domain name; other times, it is used to mean the most specific subname of a fully qualified domain name. For example, if mycomputer.city.company.com is the fully qualified domain name, either of the following can be considered the host name: mycomputer.city.company.com, mycomputer.

hover help
Information that can be viewed by holding a mouse over an item such as an icon in the user interface.

HTTP See *Hypertext Transfer Protocol*.

HTTPS
See *Hypertext Transfer Protocol Secure*.

Hypertext Transfer Protocol (HTTP)
In the Internet suite of protocols, the protocol that is used to transfer and display hypertext and XML documents.

Hypertext Transfer Protocol Secure (HTTPS)
A TCP/IP protocol that is used by World Wide Web servers and Web browsers to transfer and display hypermedia documents securely across the Internet.

I

ID data
An ID data structure holds an individual result from a statistical API function.

identity propagation
The concept of preserving a user's security identity information (the distributed identity) independent of where the identity information has been created, for use during authorization and for auditing purposes. The distributed identity is carried with a request from the distributed client application to the CICS server, and is incorporated in the access control of the server as part of the authorization process, for example, using RACF. CICS Transaction Gateway flows the distributed identity to CICS. See also *distributed identity*.

identity propagation login module
A code component that provides support for identity propagation. The identity propagation login module is included with the CICS Transaction

Gateway ECI resource adapter (cicseci.rar), conforms to the JAAS specification and is contained in a single Java class within the resource adapter. See also *identity propagation*.

iKeyman
A tool for maintaining digital certificates for JSSE.

in doubt
The state of a transaction that has completed the prepare phase of the two-phase commit process and is waiting to be completed.

in flight
The state of a transaction that has not yet completed the prepare phase of the two-phase commit process.

independent logical unit
A logical unit (LU) that can both send and receive a BIND, and which supports single, parallel, and multiple sessions. See *BIND*.

<install_path>
This term is used in file paths to represent the directory where you installed the product.

Internet Architecture Board
The technical body that oversees the development of the internet suite of protocols known as TCP/IP.

Internet Protocol (IP)
In TCP/IP, a protocol that routes data from its source to its destination in an Internet environment.

interoperability
The capability to communicate, run programs, or transfer data among various functional units in a way that requires the user to have little or no knowledge of the unique characteristics of those units.

IP Internet Protocol.

IPIC See *IP interconnectivity*.

IP address
A unique address for a device or logical unit on a network that uses the IP standard.

IP interconnectivity (IPIC)
The IPIC protocol enables Distributed Program Link (DPL) access from a non-CICS program to a CICS program over TCP/IP, using the External Call Interface (ECI). IPIC passes and receives data using COMMAREAs, or containers.

J

JEE (formerly J2EE)
See *Java 2 Platform Enterprise Edition*

JEE Connector architecture (JCA)
A standard architecture for connecting the JEE platform to heterogeneous enterprise information systems (EIS).

Java An object-oriented programming language for portable interpretive code that supports interaction among remote objects.

Java 2 Platform Enterprise Edition (JEE)
An environment for developing and deploying enterprise applications,

defined by Sun Microsystems Inc. The JEE platform consists of a set of services, application programming interfaces (APIs), and protocols that allow multi-tiered, Web-based applications to be developed.

JavaBeans
As defined for Java by Sun Microsystems, a portable, platform-independent, reusable component model.

Java Client application
The Java client application is a user application written in Java, including servlets and enterprise beans, that uses the Gateway classes.

Java Development Kit (JDK)
The name of the software development kit that Sun Microsystems provided for the Java platform, up to and including v 1.1.x. Sometimes used erroneously to mean the Java platform or as a generic term for any software developer kits for Java.

JavaGateway
The URL of the CICS Transaction Gateway with which the Java Client application communicates. The JavaGateway takes the form `protocol://address:port`. These protocols are supported: `tcp://`, `ssl://`, and `local:`. CICS Transaction Gateway runs with the default port value of 2006. This parameter is not relevant if you are using the protocol `local:`. For example, you might specify a JavaGateway of `tcp://ctg.business.com:2006`. If you specify the protocol as `local:` you will connect directly to the CICS server, bypassing any CICS Transaction Gateway servers.

Java Native Interface (JNI)
A programming interface that allows Java code running in a Java virtual machine to work with functions that are written in other programming languages.

Java Runtime Environment (JRE)
A subset of the Java Software Development Kit (SDK) that supports the execution, but not the development, of Java applications. The JRE comprises the Java Virtual Machine (JVM), the core classes, and supporting files.

Java Secure Socket Extension (JSSE)
A Java package that enables secure Internet communications. It implements a Java version of the Secure Sockets Layer (SSL) and Transport Layer Security (TSL) protocols and supports data encryption, server authentication, message integrity, and optionally client authentication.

Java virtual machine (JVM)
A software implementation of a processor that runs compiled Java code (applets and applications).

JDK See *Java development kit*.

JCA See *JEE Connector Architecture* .

JNI See *Java Native Interface*.

JRE See *Java Runtime Environment*

JSSE See *Java Secure Socket Extension*.

JVM See *Java Virtual Machine*.

K

keyboard mapping
A list that establishes a correspondence between keys on the keyboard and characters displayed on a display screen, or action taken by a program, when that key is pressed.

Keystore
In the JSSE protocol, a file that contains public keys, private keys, trusted roots, and certificates.

L

local mode
Local mode describes the use of the CICS Transaction Gateway *local* protocol. The Gateway daemon is not used in local mode.

local transaction
A recoverable unit of work managed by a resource manager and not coordinated by an external transaction manager.

logical CICS server
An alias that can be passed on an ECI request when running in remote mode to CICS Transaction Gateway for z/OS. The alias name is mapped to an actual CICS server name by a dynamic server selection (DSS) mechanism.

logical end of day
The local time of day on the 24-hour clock to which a Gateway daemon aligns statistics intervals. If the statistics interval is 24 hours, this is the local time at which interval statistics will be reset and, on z/OS, optionally recorded to SMF. This time is set using the **stateod** parameter in the configuration file (ctg.ini).

logical unit (LU)
In SNA, a port through which an end user accesses the SNA network to communicate with another end user and through which the end user accesses the functions provided by system services control points (SSCP). An LU can support at least two sessions, one with an SSCP and one with another LU, and might be capable of supporting many sessions with other logical units. See also *network addressable unit*, *primary logical unit*, *secondary logical unit*.

logical unit 6.2 (LU 6.2)
A type of logical unit that supports general communications between programs in a distributed processing environment.

The LU type that supports sessions between two applications using APPC.

logical unit of work (LUW)
The processing that a program performs between synchronization points

LU See *logical unit*.

LU 6.2 See *logical unit 6.2*.

LU-LU session
In SNA, a session between two logical units (LUs) in an SNA network. It provides communication between two end users, or between an end user and an LU services component.

LU-LU session type 6.2
In SNA, a type of session for communication between peer systems. Synonymous with APPC protocol.

LUW See *logical unit of work*.

M

managed mode
Describes an environment in which connections are obtained from connection factories that the JEE server has set up. Such connections are owned by the JEE server.

media access control (MAC) sublayer
One of two sublayers of the ISO Open Systems Interconnection data link layer proposed for local area networks by the IEEE Project 802 Committee on Local Area Networks and the European Computer Manufacturers Association (ECMA). It provides functions that depend on the topology of the network and uses services of the physical layer to provide services to the logical link control (LLC) sublayer. The OSI data link layer corresponds to the SNA data link control layer.

method
In object-oriented programming, an operation that an object can perform. An object can have many methods.

mode In SNA, a set of parameters that defines the characteristics of a session between two LUs.

N

name server
In TCP/IP, synonym for Domain Name Server. In Internet communications, a host that translates symbolic names assigned to networks and hosts into IP addresses.

NAU See *network addressable unit*.

network address
In SNA, an address, consisting of subarea and element fields, that identifies a link, link station, or network addressable unit (NAU). Subarea nodes use network addresses; peripheral nodes use local addresses. The boundary function in the subarea node to which a peripheral node is attached transforms local addresses to network addresses and vice versa. See also *network name*.

network addressable unit (NAU)
In SNA, a logical unit, a physical unit, or a system services control point. The NAU is the origin or the destination of information transmitted by the path control network. See also *logical unit, network address, network name*.

network name
In SNA, the symbolic identifier by which end users refer to a network addressable unit (NAU), link station, or link. See also *network address*.

node type
In SNA, a designation of a node according to the protocols it supports and the network addressable units (NAUs) it can contain. Four types are defined: 1, 2, 4, and 5. Type 1 and type 2 nodes are peripheral nodes; type 4 and type 5 nodes are subarea nodes.

nonextended logical unit of work
See *SYNCONRETURN*.

nonmanaged mode
An environment in which the application is responsible for generating and

configuring connection factories. The JEE server does not own or know about these connection factories and therefore provides no Quality of Service facilities.

O

object In object-oriented programming, a concrete realization of a class that consists of data and the operations associated with that data.

object-oriented (OO)
Describing a computer system or programming language that supports objects.

one-phase commit
A protocol with a single commit phase, that is used for the coordination of changes to recoverable resources when a single resource manager is involved.

OO See *object-oriented*.

P

pacing
A technique by which a receiving station controls the rate of transmission of a sending station to prevent overrun.

parallel session
In SNA, two or more concurrently active sessions between the same two LUs using different pairs of network addresses. Each session can have independent session parameters.

PING In Internet communications, a program used in TCP/IP networks to test the ability to reach destinations by sending the destinations an Internet Control Message Protocol (ICMP) echo request and waiting for a reply.

partner logical unit (PLU)
In SNA, the remote participant in a session.

partner transaction program
The transaction program engaged in an APPC conversation with a local transaction program.

password phrase
A character string, between 9 and 100 characters in length, that is used for authentication when a user signs on to CICS. Because a password phrase can provide an exponentially greater number of possible combinations of characters than a standard 8 character password, the use of password phrases can enhance system security. Password phrases are verified by the External Security Manager (ESM), and can contain alphanumeric characters, and any of the other non alphanumeric characters that are supported by the ESM. See also *External Security Manager (ESM)*.

PLU See *primary logical unit* and *partner logical unit*.

port An endpoint for communication between devices, generally referring to a logical connection. A 16-bit number identifying a particular Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) resource within a given TCP/IP node.

port sharing
A way of load balancing TCP/IP connections across a group of servers running in the same z/OS image.

prepare phase
The first phase of a XA process in which all participants are requested to confirm readiness to commit.

presentation logic
The part of a distributed application that is concerned with the user interface of the application. Compare with *business logic*.

primary logical unit (PLU)
In SNA, the logical unit that contains the primary half-session for a particular logical unit-to-logical unit (LU-to-LU) session. See also *secondary logical unit*.

protocol boundary
The signals and rules governing interactions between two components within a node.

Q

Query strings
Query strings are used in the statistical data API. A query string is an input parameter, specifying the statistical data to be retrieved.

R

RACF See *Resource Access Control Facility*.

realm A named collection of users and groups that can be used in a specific security context. See also *distinguished name* and *identity propagation*.

Recoverable resource management services (RRMS)
The registration services, context services, and resource recovery services provided by the z/OS sync point manager that enable consistent changes to be made to multiple protected resources.

Resource Access Control Facility (RACF)
An IBM licensed program that provides access control by identifying users to the system; verifying users of the system; authorizing access to protected resources; logging detected unauthorized attempts to enter the system; and logging detected accesses to protected resources.

region In workload management on CICS Transaction Gateway for Windows, an instance of a CICS server.

remote mode
Remote mode describes the use of one of the supported CICS Transaction Gateway network protocols to connect to the Gateway daemon.

remote procedure call (RPC)
A protocol that allows a program on a client computer to run a program on a server.

Request monitoring exits
Exits that provide information about individual requests as they are processed by the CICS Transaction Gateway.

request unit (RU)
In SNA, a message unit that contains control information such as a request code, or function management (FM) headers, end-user data, or both.

request/response unit
A generic term for a request unit or a response unit. See also *request unit* and *response unit*.

response file
A file that contains predefined values that is used instead of someone having to enter those values one at a time. See also *CID methodology*.

response unit (RU)
A message unit that acknowledges a request unit; it can contain prefix information received in a request unit.

Resource adapter
A system-level software driver that is used by an EJB container or an application client to connect to an enterprise information system (EIS). A resource adapter plugs in to a container; the application components deployed on the container then use the client API (exposed by adapter) or tool-generated, high-level abstractions to access the underlying EIS.

resource group ID
A resource group ID is a logical grouping of resources, grouped for statistical purposes. A resource group ID is associated with a number of resource group statistics, each identified by a statistic ID.

resource ID
A resource ID refers to a specific resource. Information about the resource is included in resource-specific statistics. Each statistic is identified by a statistic ID.

resource manager
The participant in a transaction responsible for controlling access to recoverable resources. In terms of the CICS resource adapters this is represented by an instance of a ConnectionFactory.

Resource Recovery Services (RRS)
A z/OS facility that provides two-phase sync point support across participating resource managers.

Result set
A result set is a set of data calculated or recorded by a statistical API function.

Result set token
A result set token is a reference to the set of results returned by a statistical API function.

rollback
An operation in a transaction that reverses all the changes made during the unit of work. After the operation is complete, the unit of work is finished. Also known as a backout.

RU See *Request unit* and *Response unit*.

RPC See *remote procedure call*.

RRMS
See *Recoverable resource management services*.

RRS See *Resource Recovery Services*.

S

SBCS See *single-byte character set*.

secondary logical unit (SLU)
In SNA, the logical unit (LU) that contains the secondary half-session for a particular LU-LU session. Contrast with primary logical unit. See also *logical unit*.

Secure Sockets Layer (SSL)
A security protocol that provides communication privacy. SSL enables client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, and message forgery. SSL applies only to internet protocols, and is not applicable to SNA.

server name remapping
See *dynamic server selection*.

servlet
A Java program that runs on a Web server and extends the server's functionality by generating dynamic content in response to Web client requests. Servlets are commonly used to connect databases to the Web.

session limit
In SNA, the maximum number of concurrently active logical unit to logical unit (LU-to-LU) sessions that a particular logical unit (LU) can support.

silent installation
Installation that does not display messages or windows during its progress. Silent installation is not a synonym of "unattended installation", although it is often improperly used as such.

single-byte character set (SBCS)
A character set in which each character is represented by 1 byte. Contrast with double-byte character set.

sign-on capable terminal
A sign-on capable terminal allows sign-on transactions that are either supplied with CICS (CESN) or written by the user, to be run. Contrast with sign-on incapable terminal.

SIT See *system initialization table*.

SLU See *secondary logical unit*.

SMIT See *System Management Interface Tool*.

SNA See *Systems Network Architecture*.

SNA sense data
An SNA-defined encoding of error information In SNA, the data sent with a negative response, indicating the reason for the response.

SNASVCMG mode name
The SNA service manager mode name. This is the architecturally-defined mode name identifying sessions on which CNOS is exchanged. Most APPC-providing products predefine SNASVCMG sessions.

socket A network communication concept, typically representing a point of connection between a client and a server. A TCP/IP socket will normally combine a host name or IP address, and a port number.

SSL See *Secure Sockets Layer*.

SSLight
An implementation of SSL, written in Java, and no longer supported by CICS Transaction Gateway.

statistic data
A statistic data structure holds individual statistical result returned after calling a statistical API function.

statistic group
A generic term for a collection of statistic IDs.

statistic ID
A label referring to a specific statistic. A statistic ID is used to retrieve specific statistical data, and always has a direct relationship with a statistic group.

standard error
In many workstation-based operating systems, the output stream to which error messages or diagnostic messages are sent.

subnet
An interconnected, but independent segment of a network that is identified by its Internet Protocol (IP) address.

subnet address
In Internet communications, an extension to the basic IP addressing scheme where a portion of the host address is interpreted as the local network address.

sync point
Synchronization point. During transaction processing, a reference point to which protected resources can be restored if a failure occurs.

SYNCONRETURN
A request where the CICS server takes a sync point on successful completion of the server program. Changes to recoverable resources made by the server program are committed or rolled-back independently of changes to recoverable resources made by the client program issuing the ECI request, or changes made by the server in any subsequent ECI request. Also referred to as a *nonextended logical unit of work*.

system initialization table (SIT)
A table containing parameters used to start a CICS control region.

System Management Command
An administrative request received by a Gateway daemon (or Gateway daemon address space on z/OS) from the **ctgadmin** command (on UNIX, Linux, or Windows) or the z/OS console. The request might be made to retrieve information about the Gateway daemon, or to alter some aspect of Gateway daemon behavior. Typically, a **ctgadmin** command in the form **ctgadmin** <command string> is entered by an operator using the command line interface, or a modify command in the form `/F <job name>,APPL=<command string>` is entered by an operator on the z/OS console.

System Management Interface Tool (SMIT)
An interface tool of the AIX operating system for installing, maintaining, configuring, and diagnosing tasks.

Systems Network Architecture (SNA)
An architecture that describes the logical structure, formats, protocols, and operational sequences for transmitting information units through the networks and also the operational sequences for controlling the configuration and operation of networks.

System SSL
An implementation of SSL, no longer supported by CICS Transaction Gateway on z/OS.

T

TCP/IP
See *Transmission Control Protocol/Internet Protocol*.

TCP/IP load balancing
The ability to distribute TCP/IP connections across target servers.

terminal emulation
The capability of a personal computer to operate as if it were a particular type of terminal linked to a processing unit and to access data. See also *emulator, emulation program*.

thread A stream of computer instructions that is in control of a process. In some operating systems, a thread is the smallest unit of operation in a process. Several threads can run concurrently, performing different jobs.

timeout
A time interval that is allotted for an event to occur or complete before operation is interrupted.

TLS See *Transport Layer Security*.

token-ring network
A local area network that connects devices in a ring topology and allows unidirectional data transmission between devices by a token-passing procedure. A device must receive a token before it can transmit data.

trace A record of the processing of a computer program. It exhibits the sequences in which the instructions were processed.

transaction manager
A software unit that coordinates the activities of resource managers by managing global transactions and coordinating the decision to commit them or roll them back.

transaction program
A program that uses the Advanced Program-to-Program Communications (APPC) application programming interface (API) to communicate with a partner application program on a remote system.

Transmission Control Protocol/Internet Protocol (TCP/IP)
An industry-standard, nonproprietary set of communications protocols that provide reliable end-to-end connections between applications over interconnected networks of different types.

Transport Layer Security (TLS)
A security protocol that provides communication privacy. TLS enables client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, and message forgery. TLS applies only to internet protocols, and is not applicable to SNA. TLS is also known as SSL 3.1.

Two-phase commit
A protocol with both a prepare and a commit phase, that is used for the coordination of changes to recoverable resources when more than one resource manager is used by a single transaction.

type 2.0 node
A node that attaches to a subarea network as a peripheral node and provides a range of end-user services but no intermediate routing services.

type 2.1 node
An SNA node that can be configured as an endpoint or intermediate routing node in a network, or as a peripheral node attached to a subarea network.

U

unattended installation
Unattended installation is installation performed without user interaction during its progress, or, with no user present at all, except for the initial launch of the process. -

Uniform Resource Locator (URL)
A sequence of characters that represent information resources on a computer or in a network such as the Internet. This sequence of characters includes (a) the abbreviated name of the protocol used to access the information resource and (b) the information used by the protocol to locate the information resource.

unit of recovery (UR)
A defined package of work to be performed by the RRS.

unit of work (UOW)
A recoverable sequence of operations performed by an application between two points of consistency. A unit of work begins when a transaction starts or at a user-requested sync point. It ends either at a user-requested sync point or at the end of a transaction.

UOW See *unit of work*.

UR See *unit of recovery*.

URL See *Uniform Resource Locator*.

user registry
The location where the distinguished name of a user is defined and authenticated. See also *distinguished name*.

user session
Any APPC session other than a SNASVCMG session.

V

verb A reserved word that expresses an action to be taken by an application programming interface (API), a compiler, or an object program.

In SNA, the general name for a transaction program's request for communication services.

version string
A character string containing version information about the statistical data API.

W

WAN See *wide area network*.

Web browser
A software program that sends requests to a Web server and displays the information that the server returns.

Web server
A software program that responds to information requests generated by Web browsers.

wide area network (WAN)
A network that provides communication services to a geographic area larger than that served by a local area network or a metropolitan area network, and that can use or provide public communication facilities.

Wrapping trace
On Windows, UNIX, and Linux, a configuration in which the **Maximum Client wrap size** setting is greater than 0. The total size of Client daemon binary trace files is limited to the value specified in the **Maximum Client wrap size** setting. With standard I/O tracing, two files, called `cicscli.bin` and `cicscli.wrp`, are used; each can be up to half the size of the **Maximum Client wrap size**.

X

XA request
Any request sent or received by the CICS Transaction Gateway in support of an XA transaction. These requests include the XA commands commit, complete, end, forget, prepare, recover, rollback, and start.

XA transaction
A global transaction that adheres to the X/Open standard for distributed transaction processing (DTP.)

Index

Special characters

Numerics

A

B

C

D

E

F

O

P

Q

R

S

T

U

V

W

X

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply in the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who want to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM United Kingdom Laboratories, MP151, Hursley Park, Winchester, Hampshire, England, SO21 2JN.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other product and service names might be trademarks of IBM or other companies.

Readers' Comments — We'd Like to Hear from You

CICS Transaction Gateway
Version 8 Release 1
UNIX and Linux Administration

Publication No. SC34-7216-00

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:
- Send your comments to the address on the reverse side of this form.
- Send a fax to the following number: +44 1962 816151
- Send your comments via email to: idrcf@uk.ibm.com

If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	Email address

Readers' Comments — We'd Like to Hear from You
SC34-7216-00

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM United Kingdom Limited
User Technologies Department (MP189)
Hursley Park
Winchester
Hampshire
United Kingdom
SO21 2JN

Fold and Tape | **Please do not staple** | Fold and Tape

SC34-7216-00

Cut or Fold
Along Line

SC34-7216-00