CICS Transaction Gateway for Multiplatforms
Version 9 Release 2

# Multiplatform Administration

SC34-7338-00

CICS Transaction Gateway for Multiplatforms
Version 9 Release 2

# Multiplatform Administration

SC34-7338-00

**Note**

**Note:** Before you use this information and the product it supports, read the information in Safety and environmental notices and Notices.

This edition applies to Version 9, Release 2 Modification 0 of CICS TG for Multiplatforms, program number 5724-I81 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 includes trademarks including Java™, for more information about Trademarks, see the Trademark information at the back of this publication.

# Part 1. What's new in CICS Transaction Gateway for Multiplatforms V9.2 and CICS Transaction Gateway Desktop Edition V9.2

CICS Transaction Gateway for Multiplatforms V9.2 and CICS Transaction Gateway Desktop Edition V9.2 include enhancements in these areas:

## CICS Intercept Plug-in

CICS TG now provides enhanced capabilities to facilitate continuous integration testing, adding the CICS Intercept plug-in (a user exit within the Gateway daemon component) as a complementary option to the existing Gateway Intercept plug-in (a user exit within the Java/JCA APIs). The plug-in can intercept ECI, EPI and ESI requests for any remote mode application, or JSON web service client, and allows selective pass-through to live CICS servers, or virtualized responses, when deploying CICS Transaction Gateway. For more information, see Chapter 35, "Configuring for application testing," on page 105 and *CICS Intercept Plug-in* in *CICS TG Developing Applications*.

## Additional platform support

In addition to SuSE and Red Hat distributions, CICS Transaction Gateway is now supported on the Ubuntu distribution for supported Intel hardware. Microsoft Windows 10 is also supported in this release.

For more information, see Chapter 10, "Supported software," on page 27

## Improved client response times over SSL connections to the Gateway daemon

CICS Transaction Gateway V9.2 delivers significant improvements in response time for remote client applications that establish a secure (TLS) connection, when many such remote clients simultaneously establish secure connections, or a large connection pool is established. This can be of assistance in start-of-business scenarios, or when entire connection pools are cycled for a planned or unplanned outage.

## Container ccsid in request monitoring exits

In previous releases, request monitoring exits provided read-only access to the channel payload of an ECI request and response, but did not provide access to the CCSID that was used for encoding containers of type `CHAR`. CICS Transaction Gateway V9.2 provides access to the CCSID for `CHAR` type containers from request monitoring exits. This can be used in association with the Gateway Intercept or CICS Intercept plug-ins to recreate the data in a channel or container. For more information, see the `getCCSID` method descriptions in ChannelInfo and ContainerInfo. You can test these methods using the Java Basic Monitor request monitoring exit sample in *CICS TG Developing Applications*.

## Running CICS Transaction Gateway on UNIX and Linux

When running the `ctgd` command it is no longer necessary to define a `ctgd.conf` configuration file. If a configuration file is not defined, the **`ctgd`** command uses the values defined in the `ctg.ini` configuration file. The **`ctgd start`** command also creates a log file `/var/cicscli/ctgd.log` if it does not exist.

The default behaviour of Gateway daemon logging has changed to use the file destination. The default location for the log and trace files is now the `/var/cicscli` directory.

The default location of the configuration, log and trace files has moved from `<install_path>/bin` to `/var/cicscli`. For more information, see "Changes to configuration files on UNIX and Linux" on page 82

## New sample EC04

A new sample COBOL program `EC04`, returns information in the COMMAREA about the CICS region where `EC04` ran and how it was called. `EC04` can be called from the Java EciB2 sample. For more information, see *Sample CICS programs and maps* in *CICS TG Developing Applications*.

## Support for new users of CICS Transaction Gateway

The CICS TG Knowledge Center contains a new chapter, *Getting started* which shows a new user how to install, configure and run CICS Transaction Gateway on Windows and to run a sample application.

For other changes from previous releases, see the Upgrading section.

# Part 2. Overview

CICS Transaction Gateway is a high-performing, secure, and scalable connector. The product uses standards-based interfaces that enable client applications deployed in various runtime environments to access CICS servers.

# Chapter 1. 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.

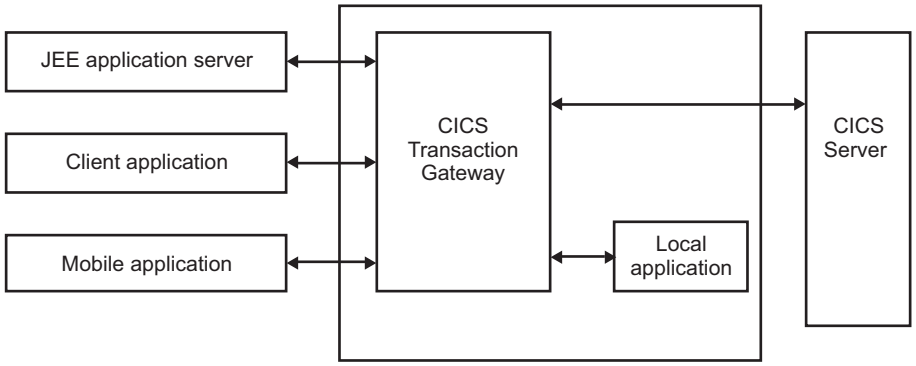



*Figure 1. Access to CICS using CICS Transaction Gateway for Multiplatforms*

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 high availability
- Access to statistics and request monitoring information
- Support for two-phase commit transactions from a JEE application server

For an explanation of the terms used in this documentation, see the Glossary.

# Chapter 2. 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.

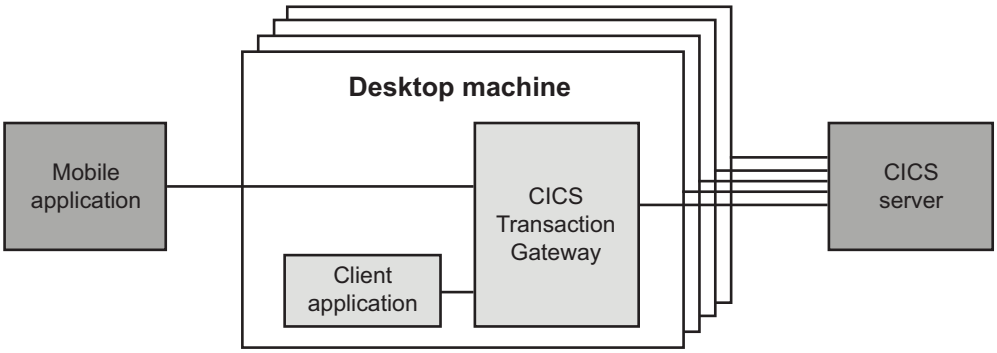



*Figure 2. Access to CICS using CICS Transaction Gateway Desktop Edition*

Client applications can be written in Java, C, C++, COM, COBOL, or the .NET Framework, 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 180
- "Maximum number of worker threads" on page 158
- "Maximum number of connection manager threads" on page 157
- "IPIC send sessions" on page 118

For an explanation of the terms used in this documentation, see the "Glossary" on page 571.

**Related information**:

Part 3, "Planning," on page 23
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.

# Chapter 3. 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, COM objects for use in local mode topologies, and a .NET Framework API for use in remote mode topologies are also included. Using these APIs, client applications can make multiple concurrent program calls to one or more CICS servers.

CICS Transaction Gateway can also expose CICS programs as JSON web services, allowing an HTTP or HTTPS client application to call CICS programs with channel and COMMAREA payloads. CICS TG does not provide an API for developing web service client applications, but can be used to develop JSON web service APIs for calling CICS programs. The behaviour of a JSON web service request has some similarities to an External Call Interface (ECI) request.

## 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*.

# Chapter 4. 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.

**Related information**:

Chapter 37, “Configuring CICS server connections,” on page 111
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.

Chapter 13, “Which protocol can be used?,” on page 39
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 can be 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 TCP, SSL, HTTP, or HTTPS.

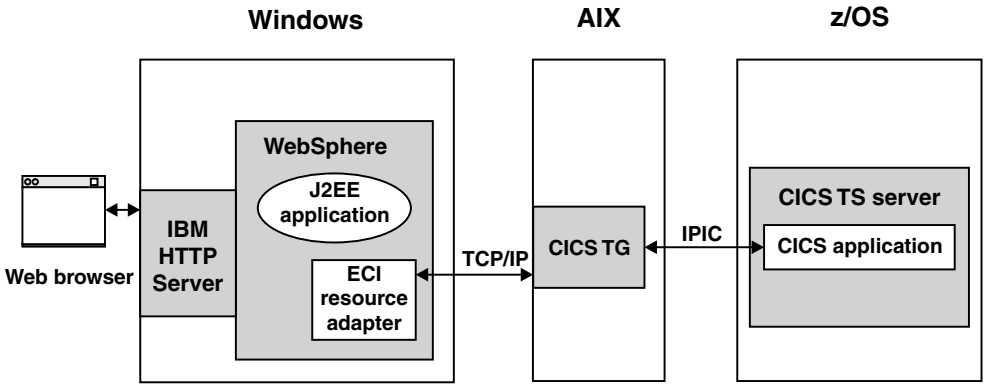



*Figure 3. An example of CICS Transaction Gateway for Multiplatforms in remote mode*

### 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

- High availability with workload balancing across multiple CICS servers
- A lightweight client footprint
- Access to statistical information
- Supports the use of applets to connect to a Gateway daemon

## Local mode

In local mode, the client application is installed and runs on the CICS Transaction Gateway host machine and sends requests directly to CICS without using the Gateway daemon.

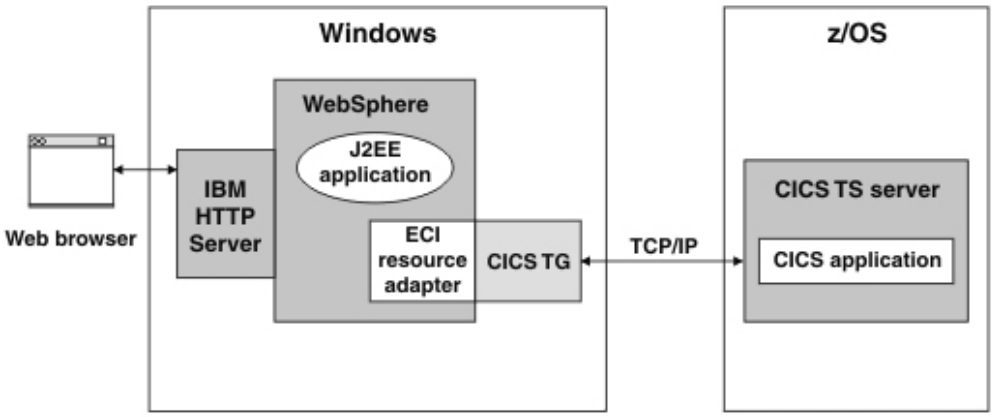



*Figure 4. An example of CICS Transaction Gateway for Multiplatforms in local mode*

### Features of local mode

Local mode is best suited to smaller scale systems and has the following features:

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

## Connectivity to CICS

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

**Note:** In these descriptions, the terms "local mode" and "remote mode" refer to whether the connection between Client application and Gateway uses a network connection. See the related topics for more information on running in local mode or remote mode.

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. IPIC also supports the offload of work to an IBM zSeries Application Assist Processor (zAAP).

**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 IBM TXSeries® servers.

# Chapter 5. 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.

## Dynamic server selection

Dynamic server selection enables the Gateway daemon to dynamically select the CICS server at run time for requests from client applications. This provides the ability to avoid a single point of failure. Client applications can be written without prior knowledge of the CICS server names at run time.

A high availability scenario can be implemented using dynamic server selection from within a user exit program. Alternatively on Windows, if using CICS servers connected with the TCP/IP or SNA protocols, the Workload Manager can be used to distribute work across multiple CICS servers.

# Chapter 6. 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
- HTTPS so that clients can connect to CICS TG and invoke a web service.
- 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.

# Chapter 7. 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.

# Chapter 8. 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 adapter 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.

# Part 3. 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 Part 5, “Upgrading,” on page 75.

# Chapter 9. Hardware requirements

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

CICS TG for Multiplatforms is supported on 64-bit operating systems only. CICS TG Desktop Edition is supported on 64-bit operating systems, and additionally on 32-bit operating systems on Intel Linux, Windows 7, Windows 8 and Windows 8.1.

On UNIX and Linux systems:

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

On Windows systems:

- An Intel Pentium, AMD Opteron or Intel EM64T system supported by Microsoft Windows.

# Chapter 10. Supported software

CICS Transaction Gateway products support various levels of IBM and third-party software.

Minimum required service levels are listed where appropriate. If a specific level is not listed, support is provided for the General Availability (GA) release of the third-party product. Service levels later than those listed are also supported, if the vendor provides upward compatibility between service releases.

**Note:** If CICS Transaction Gateway support information provided by product vendors conflicts with the information provided here, or if you experience unanticipated problems, consult your vendor's product support, and notify your IBM support representative. For end of service information see `http://www-01.ibm.com/software/support/lifecycle/index_c.html`.

## CICS Transaction Gateway core components

The platforms and runtime environments supported by the core components include the Gateway daemon, Client daemon, utilities installed with the product, applications that use the ECI V1, EPI C, ESI V1 and C APIs, and the Java API in local mode.

### Operating systems

CICS TG for Multiplatforms runs on IBM AIX, HP-UX, Linux on Intel, Linux on POWER®, Linux on z Systems, Solaris and Microsoft Windows.

CICS TG for Multiplatforms is supported on 64-bit operating systems only. CICS TG Desktop Edition is supported on 64-bit operating systems, and additionally on 32-bit operating systems on Intel Linux, Windows 7, and Windows 8.1.

#### Windows

- Windows 10 Pro and Enterprise Editions
- Windows Server 2012 R2: Foundation Edition, Essentials Edition, Standard Edition, Datacenter Edition
- Windows Server 2012: Foundation Edition, Essentials Edition, Standard Edition, Datacenter Edition
- Windows 8.1: Standard Edition, Enterprise Edition, Professional Edition
- Windows 7 service pack 1: Business Edition, Professional Edition, Enterprise Edition, Ultimate Editions

#### IBM AIX

- IBM AIX 7.2
- IBM AIX 7.1 TL 03 and later.

#### Linux on Intel

- Red Hat Enterprise Linux V7.1
- Red Hat Enterprise Linux V7.1 compatible environments
- Red Hat Enterprise Linux V6.7

- Red Hat Enterprise Linux V6.7 compatible environments
- SuSE Linux Enterprise Server 12 service pack 1 and later
- SuSE Linux Enterprise Desktop 12 service pack 1 and later
- SuSE Linux Enterprise Server 11 service pack 4 and later
- SuSE Linux Enterprise Desktop 11 service pack 4 and later
- Ubuntu 14.04.3 and later

### Linux on POWER (Big Endian)

- SuSE Linux Enterprise Server 11 service pack 4 and later
- Red Hat Enterprise Linux V7⌂.1
- Red Hat Enterprise Linux V6⌂.7

### Linux on IBM z Systems

- SuSE Linux Enterprise Server 12 service pack 1 and later (See Note)
- SuSE Linux Enterprise Server 11 service pack 4 and later
- Red Hat Enterprise Linux V7.1
- Red Hat Enterprise Linux V6.7⌂

**Note:** SuSE Linux Enterprise Server 12 was not generally available when Communications Server for Data Center Deployment v7.0.0.2 was released, so SNA on SuSE Linux Enterprise Server 12 is not supported. Support will be provided in a future APAR (fix release) LI78082. You can follow the support page for CSDCD at: http://www.ibm.com/support/entry/portal/product/other_software/communications_server_for_data_center_deployment.

### Solaris on SPARC

- Solaris 11
- Solaris 10

### HP-UX on Itanium

- HP-UX 11i V3⌂⌂

**Note:**

1. CICS Transaction Gateway is a 32-bit application.
2. The setting of multiple user interface languages on Windows server platforms is not supported.
3. Multiple concurrent users using different user interface languages are not supported. This situation can occur when using fast user switching.
4. All UNIX and Linux operating systems require the Korn shell to be installed before you install CICS Transaction Gateway.
5. CICS Transaction Gateway does not support Security-Enhanced Linux.
6. Red Hat Enterprise Linux compatible environments must be binary and source compatible with supported Red Hat Enterprise Linux versions listed above.

## Java runtime environments for core components

The Gateway daemon and local mode applications are supported on a number or Java runtime environments.

### IBM AIX

IBM 32-bit Runtime Environment for IBM AIX, Java Technology Edition, Version 8.0 SR2 FP10, and later service refreshes

### HP-UX on Itanium

HP 32-bit Runtime Environment for JSE HP-UX 11i platform, adapted by IBM for IBM software, Version 8.0 SR2, and later service refreshes. Some HP-UX patches may be required for this version of Java. Refer to the HP Support Center for more details.

### Linux on Intel

IBM 32-bit Runtime Environment for Linux on Intel architecture, Java Technology Edition, Version 8.0 SR2 FP10, and later service refreshes.

### Linux on POWER

IBM 32-bit Runtime Environment for Linux on System p architecture, Java Technology Edition, Version 8.0 SR2 FP10, and later service refreshes.

### Linux on IBM z Systems

IBM 31-bit Runtime Environment for Linux on IBM z Systems architecture, Java Technology Edition, Version 8.0 SR2 FP10, and later service refreshes.

### Solaris on SPARC

IBM 32-bit Runtime Environment for Solaris, Java Technology Edition, Version 7.0 SR9 FP30, and later service refreshes.

### Windows

IBM 32-bit Runtime Environment for Windows, Java Technology Edition, Version 8.0 SR2 FP10, and later service refreshes

## CICS servers

CICS Transaction Gateway can connect to a number of CICS servers.

- CICS Transaction Server for IBM z/OS® V4.1 and later including CICS Transaction Server for IBM z/OS Value Unit Edition and CICS Transaction Server for IBM z/OS Developer Trial.
- CICS Transaction Server for z/VSE® V2.1 and later.
- CICS Transaction Server for VSE/ESA V1.1.1 and later.
- CICS Transaction Server for i V7.1 and later.
- IBM TXSeries for Multiplatforms V8.1⌂ and later.
- IBM TXSeries for Multiplatforms V7.1⌂ and later.
- CICS Transaction Server for IBM z/OS Developer Trial.
- CICS Transaction Server for IBM z/OS Value Unit Edition.

## SNA communication products

CICS Transaction Gateway supports a number of SNA servers.

CICS Transaction Gateway requires 32-bit SNA API libraries. Some SNA server products might install 64-bit SNA API libraries on 64-bit platforms by default. For more information on how to install 32-bit SNA API libraries on 64-bit platforms, see your SNA server documentation.

### IBM AIX

- IBM Communications Server for Data Center Deployment V7.0
- IBM Communications Server for IBM AIX V6.4

### HP-UX

- SNAplus2 R7

### Linux on Intel

- IBM Communications Server for Data Center Deployment V7.0
- IBM Communications Server for Linux V6.4.0

### Linux on POWER

- IBM Communications Server for Data Center Deployment V7.0
- IBM Communications Server for Linux V6.4.0

### Linux on IBM z Systems

- IBM Communications Server for Data Center Deployment V7.0
- IBM Communications Server for Linux on IBM zSeries V6.4.0

### Solaris

- SNAP-IX V7 for Solaris

### Windows

- IBM Communications Server for Data Center Deployment V7.0 (using the Windows Remote API Client)
- IBM Communications Server for Windows V6.4.0
- IBM Personal Communications V6.0
- Microsoft Host Integration Server 2010
- Microsoft Host Integration Server 2013

**Note:** If you are using Microsoft Host Integration Server, you might need to configure the CICS Transaction Gateway with a Windows domain logon with the appropriate access to the HIS server.

## JEE application servers

The CICS Transaction Gateway JEE resource adapters are compatible with a number of JEE application servers.

**Note:** This information does not apply to CICS Transaction Gateway Desktop Edition.

### IBM WebSphere Application Server for Multiplatforms⌂ V8.0 and V8.5

Supported in local and remote modes. If you are using a 64-bit application server to connect to CICS in a local mode topology, you must use IPIC on connections to CICS.

### IBM WebSphere Application Server for z/OS V8.0 and V8.5

Supported in local and remote modes.

### IBM WebSphere Application Server for Multiplatforms⌂ V7.0

Supported in remote mode only⌂, using the CICS Transaction Gateway V8.0 JCA 1.5 resource adapters connecting to CICS Transaction Gateway V9.2⌂. See Note 1.

### IBM WebSphere Application Server for z/OS V7.0

Supported in remote mode only⌂, using the CICS Transaction Gateway V8.0 JCA 1.5 resource adapters connecting to CICS Transaction Gateway V9.2. See Note 1.

### JEE 7 application servers

Any JEE 7 or above certified application server that successfully runs the JCA resource adapter IVT (installation verification test). ⌂This includes all non-IBM application servers such as Oracle WebLogic, Glassfish, JBoss and others. The IVT provided with CICS Transaction Gateway must run successfully before problems can be reported to IBM. If you are using a 64-bit application server to connect to CICS in a local mode topology, you must use IPIC on connections to CICS.

### JEE 6 application servers

Any JEE 6 or above certified application server that successfully runs the JCA resource adapter IVT (installation verification test). ⌂This includes all non-IBM application servers such as Oracle WebLogic, Glassfish, JBoss and others. The IVT test provided with CICS Transaction Gateway must run successfully before problems can be reported to IBM. If you are using a 64-bit application server to connect to CICS in a local mode topology, you must use IPIC to connect to CICS.⌂

**Note:**

1. The CICS Transaction Gateway V8.0 resource adapters are available for download in ⌂SupportPac CC03⌂.⌂ For more information see http://www-01.ibm.com/support/docview.wss?uid=swg24008817
2. For more information about the JCA resource adapter IVT see “JCA resource adapter installation verification test (IVT)” on page 251.

## Applications

CICS Transaction Gateway supports a range of environments for your applications.

## Java runtime environments for applications

A number of Java Runtime Environments (JRE) support remote mode applications that use the CICS Transaction Gateway Java or JEE APIs.

### Java SE 8

All IBM provided Java SE environments are supported. ⌂ Any Java 8 compatible SDK carrying the Java Compatible logo is supported.⌂

### Java SE 7

All IBM provided Java SE environments are supported. ⌂ Any Java 7 compatible SDK carrying the Java Compatible logo is supported.⌂

### Java SE 6

All IBM provided Java SE environments are supported. ⌂ Any Java 6 compatible SDK carrying the Java Compatible logo is supported.⌂

**Note:**

1. 32-bit (31-bit on Linux on zSeries) or 64-bit Java Runtime Environments can be used.
2. Supports the JCA resource adapters when used nonmanaged.
3. Use the latest Java update for your Java Runtime Environment (JRE).

## Operating systems for applications

A number of operating systems can be used with CICS Transaction Gateway application run time components available in the CICS TG SDK.

See the list of supported operating systems in "Operating systems" on page 27.

## Development environments

The CICS Transaction Gateway APIs can be used with the specified application development tools. Application development frameworks built upon the specified tools are also supported.

### Windows

- Microsoft Visual Studio 2015⌂
- Microsoft Visual Studio 2013⌂
- Microsoft .NET Framework 4.6
- Microsoft .NET Framework 4.5
- Microsoft .NET Framework 4.0
- Microsoft .NET Framework 3.5

### IBM AIX

- XL C/C++ for IBM AIX V13.1 and later
- XL C/C++ for IBM AIX V12.1
- XL C/C++ for IBM AIX V11.1⌂

### Solaris

- Oracle Solaris Studio 12.3

### HP-UX

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

### Linux

- XL C/C++ for Linux, V13.1 and later
- XL C/C++ for Linux, V12.1
- XL C/C++ for Linux, V11.1⌂
- gcc 4.3 compatible runtime⌂ for Linux

### COBOL compilers

Any 32-bit ANSI compliant COBOL compiler is supported for use with the product C APIs.

- IBM COBOL for IBM AIX V5.1 and later
- IBM COBOL for IBM AIX V4.1 and later
- OEM ANSI compliant COBOL compiler⌂

### Java

The following IBM provided Java SE environments are supported:

- Java SE 8⌂
- Java SE 7⌂
- Java SE 6⌂

Any Java 8, Java 7, or Java 6 compatible SDK carrying the Java Compatible logo is supported.

**Note:**

1. Development using the .NET Framework API is supported only on Windows.
2. C++ and COBOL environments are supported for developing 32-bit applications only.
3. C environments are supported for developing 64-bit applications only with the ECI and ESI V2 APIs.
4. Use the latest Java update for your Java Run-time Environment (JRE).
5. Applications can be developed on any supported run-time platforms.⌂

## CICS Explorer and z/OS Explorer

CICS Transaction Gateway includes a plug-in that can be used in either CICS Explorer or z/OS Explorer.

- CICS Transaction Gateway plug-in V3.0.1 is supported in CICS Explorer V5.3.0.
- CICS Transaction Gateway plug-in for CICS Explorer V2.1 and V3.0 are supported in CICS Explorer V5.2.0.

You can use any version of the CICS Transaction Gateway plug-in with CICS Transaction Gateway. However, to access all the latest features, you should use the latest versions of the CICS Explorer or z/OS Explorer and CICS Transaction Gateway plug-in.

# Chapter 11. 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 IBM z/OS or Linux operating systems.
- IBM PowerVM® hypervisor with IBM 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.
- SUSE KVM with SUSE Linux Enterprise Server (SLES) operating system.

If using a virtualization environment, defect support is available when the operating system and any additional software used by CICS TG is listed in the supported software statement for the version of CICS TG that you are using.

Unless stated otherwise, CICS TG has not been tested in specific virtualization environments. CICS TG Support is therefore unable to assist in issues related to configuration and setup, or issues directly related to, the virtualization environment itself. If issues arise that are related to the virtualization environment, the user may need to contact the virtualization environment vendor for support, or the issue may need to be recreated outside of the virtualization environment in order to receive CICS TG support.

This statement also applies to shared root configurations, such as IBM AIX WPARs and Solaris Zones. CICS TG has not been adapted to exploit shared root virtualization, so some configurations may not be achievable.

## Operating system-based virtualization solutions

If CICS Transaction Gateway for Multiplatforms is running on IBM 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 “Operating systems” on page 27.

# Chapter 12. Dynamic logical partitioning on IBM 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.

# Chapter 13. 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 Chapter 14, "Which API can be used?," on page 41.

*Table 1. Supported connectivity scenarios*

| Header | IPIC IPv6 | IPIC IPv4 | TCP/IP IPv6 | TCP/IP IPv4 | SNA |
|---|---|---|---|---|---|
| CICS Transaction Server for IBM z/OS V5.3 | XA, ECI, channels, ESI, password phrases | XA, ECI, channels, ESI, password phrases | ECI | ECI | ECI, EPI, ESI |
| CICS Transaction Server for IBM z/OS V5.2 | XA, ECI, channels, ESI, password phrases | XA, ECI, channels, ESI, password phrases | ECI | ECI | ECI, EPI, ESI |
| CICS Transaction Server for IBM z/OS V5.1 | XA, ECI, channels, ESI, password phrases | XA, ECI, channels, ESI, password phrases | ECI | ECI | ECI, EPI, ESI |
| CICS Transaction Server for IBM z/OS V4.2 | XA, ECI, channels, ESI, password phrases | XA, ECI, channels, ESI, password phrases | ECI | ECI | ECI, EPI, ESI |
| CICS Transaction Server for IBM z/OS V4.1 | XA, ECI, channels, ESI | XA, ECI, channels, ESI | ECI | ECI | ECI, EPI, ESI |
| CICS Transaction Server for VSE V1.1.1 and CICS/VSE V2.3 | Not supported | Not supported | Not supported | ECI | ECI, EPI, ESI |
| IBM TXSeries V8.2 | Not supported | ECI, channels | Not supported | ECI, EPI | Not supported |
| IBM TXSeries V8.1 | Not supported | ECI, channels | Not supported | ECI, EPI | Not supported |
| CICS Transaction Server for i V7.1 | Not supported | Not supported | ECI, EPI | ECI, EPI | ECI, EPI, ESI |

**Note:**

- ECI denotes ECI COMMAREA application support.
- EPI denotes EPI API, CICS 3270 terminal emulator and CICS 3270 terminal printer support.
- XA is supported in local mode only.

# Chapter 14. Which API can be used?

This table shows which APIs are supported over the IPIC, TCP/IP and SNA CICS server connection 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 Chapter 13, "Which protocol can be used?," on page 39

| API | IPIC local mode | IPIC remote mode | TCP/IP local mode (32 bit only) | TCP/IP remote mode | SNA local mode (32 bit only) | SNA remote mode |
|---|---|---|---|---|---|---|
| Java ECI | ✓ 1 | ✓ 1 | ✓ | ✓ | ✓ | ✓ |
| Java ESI | ✓ | ✓ | x | x | ✓ | ✓ |
| Java EPI | x | x | ✓ | ✓ | ✓ | ✓ |
| JEE non-XA | ✓ 1 | ✓ 1 | ✓ | ✓ | ✓ | ✓ |
| JEE XA | ✓ 1 | x | x | x | x | x |
| JSON web services | x | ✓ 1 | x | ✓ | x | ✓ |
| C/C++/ COBOL ECI V1 | x | x | ✓ | x | ✓ | x |
| C/C++/ COBOL ESI V1 | x | x | x | x | ✓ | x |
| C/C++/ COBOL EPI | x | x | ✓ | x | ✓ | x |
| C ECI V2 | x | ✓ 1 | x | ✓ | x | ✓ |
| C ESI V2 | x | ✓ | x | x | x | ✓ |
| .NET ECI | x | ✓ 1 | x | ✓ | x | ✓ |
| .NET ESI | x | ✓ | x | x | x | ✓ |
| COM ECI | x | x | ✓ | x | ✓ | x |
| COM EPI | x | x | ✓ | x | ✓ | x |

**Note:**

**1** Includes channel application support. All ECI APIs and protocols support COMMAREA based applications.

# Chapter 15. 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.

## Application compatibility

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

### NET Framework-based application compatibility

Compatibility of Microsoft NET Framework-based applications with different versions of the CICS Transaction Gateway .NET Framework API.

Client applications that were built with an earlier version of the CICS Transaction Gateway .NET Framework API can connect to a V9.2 Gateway daemon by using the CICS Transaction Gateway .NET Framework API assembly that they were compiled against. You can also configure existing applications to use the CICS Transaction Gateway V9.2 .NET Framework API by using assembly redirection. For more information, see “Assembly redirection for Microsoft NET Framework-based applications” on page 85.

If the application was compiled against the CICS Transaction Gateway V8.0 .NET Framework API, assembly redirection might not be possible and the behavior of some parts of the API is changed. For more information, see “Removed dependency on ECI V2 for Microsoft NET Framework-based applications” on page 86.

### 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 and ESI 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 and ESI 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, if the following conditions are met:

- The exit programs execute with the version of Java required by CICS Transaction Gateway.
- There is no specific requirement to do so for the version of the product from which you are upgrading. For more information see *Upgrading*.

# Resource adapter compatibility

CICS Transaction Gateway can be used with earlier versions of the resource adapters.

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 that are still in support.
- You cannot use a remote Gateway daemon with later version resource adapters.
- When using local mode, the resource adapter and CICS Transaction Gateway versions must be the same.
- 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.
- The upward compatability for the resource adapters is maintained, so that older resource adapters work with newer Gateway daemons, but after a release of CICS TG is out of support, no compatability testing is done for the out of support resource adapters.

**Related information**:

"JCA resource adapter installation verification test (IVT)" on page 251
The JCA resource adapter installation verification test (IVT) verifies whether the ECI resource adapter can be used with a particular application server.

# Chapter 16. National Language Support

CICS Transaction Gateway supports: English, French, German, Italian, Japanese, Korean, Simplified Chinese, Spanish, and Turkish.

For information about code pages see “Changing the system locale” on page 96.

# Chapter 17. Code page support

Information about the code page support provided for CICS Transaction Gateway 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 Part 16, "Data conversion," on page 553, 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 IBM 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) | IBM 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) | IBM 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) | IBM 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) | IBM AIX | 1381 |
| zh_CN (1383) | HP | 1381 |
| zh_CN (1383) | Linux | 1381 |

| CICS Client code page | CICS Server operating system | CICS Server code page |
|---|---|---|
| zh_CN (1383) | Solaris | 1381 |
| zh_CN (1383) | Windows | 1381 |

# Part 4. Installing

To install CICS Transaction Gateway follow the steps in these topics.

# Chapter 18. Product versions

The version of CICS Transaction Gateway is defined as a string comprising four numbers separated by dots. The first two numbers are used on their own as an abbreviated product version, the last two numbers identify the modification level and fix pack.

To obtain the number of the installed version of CICS Transaction Gateway, run the command: **cicscli -v**

An example of the messages returned by running the **cicscli -v** command on IBM AIX:

```
CCL8001I CICSCLI - CICS Transaction Gateway Client Daemon Control Program
CCL0002I (C) Copyright IBM Corporation 1994, 2015.  All rights reserved.
CCL8029I CICS Transaction Gateway for Multiplatforms for AIX Version 9.2.0.0
CCL8074I Build Level 'c920-20150121-0205', 'GA'
CCL8023I CICSCLI performed no action
```

Message CCL8029I includes “Version 9.2.0.0”; this shows that the product version is “9.2” with Fix Pack 0. When the modification level is “0” it is normally ignored.

Message CCL8074I includes “GA”, this shows the CICS Transaction Gateway installation was installed by a full product install. When message CCL8074I includes “FP” instead of “GA”, the string “FP” shows that a fix pack has been installed after a full product install.

# Chapter 19. Preparing to install CICS Transaction Gateway

These tasks must be completed before attempting to install CICS Transaction Gateway.

1. Check that your operating system is supported; for more information see "Operating systems" on page 27.
2. Only one of the products CICS TG for Multiplatforms or CICS Transaction Gateway Desktop Edition can be installed on an operating system at a time. Any version of any of these products, including a beta release, must be removed before you install another version of one of these products.
3. Product upgrade support is as follows:
   - CICS Transaction Gateway for Multiplatforms can be upgraded from V7.0 or later. Earlier and beta versions must be removed first.
   - CICS Transaction Gateway Desktop Edition can be upgraded from V8.0 or later.
4. When a CICS Transaction Gateway installer is named with a version string that includes a fix pack number then that installer includes all the product updates supplied by that fix pack. For more information about version strings see Product versions.

## Disk space requirements

The following table shows the disk space requirements for CICS Transaction Gateway. Additional space is required for log and trace data. During installation, extra space is required but this is freed when installation is complete:

*Table 2.* . Disk space requirements for CICS TG

| Disk space requirement | Windows | Solaris | AIX | Linux | HP |
|---|---|---|---|---|---|
| Product | 380 MB | 370 MB | 360 MB | 330 MB | 410 MB |
| Additional temporary space needed for new installation | 650 MB | 600 MB | 600 MB | 600 MB | 600 MB |
| Additional temporary space needed for upgrade | 750 MB | 700 MB | 700 MB | 700 MB | 700 MB |

## Windows platforms

To prepare to install CICS Transaction Gateway:

1. Log on with administrator privileges. Administrator authority is required for installing and uninstalling. Administrator privileges are available if you are a member of the local Administrators group or if you are a domain Administrator.
2. Check that there is sufficient free disk space.
3. If you are upgrading, stop any CICS Transaction Gateway application processes, product processes or services that are running, for example the IBM CICS Transaction Gateway service and the Configuration Tool.
4. When using Windows Server Remote Desktop Services, ensure that your session is correctly configured to install new software. Take one of the following steps:

- Install from **Add or Remove Programs** on the Control Panel.
- At a command prompt, type `change user /install` to place the system into install mode. When the installation has finished, type `change user /execute` at the command prompt to return the system to execute mode.

## UNIX and Linux platforms

Check the table for the disk space requirements for the installation after the install process finishes. Additional space is required for log and trace data. To successfully complete an installation ensure that additional temporary disk space is available; 460 MB is required for a new installation and 690 MB is required for an upgrade installation.

To prepare to install CICS Transaction Gateway:

1. Log on as root.
2. The following libraries must be installed before you run the UNIX/Linux CICS TG installer in the default Graphical mode:
   - `libXtst` - X11 Resource extension library
   - `libXft` - Font drawing library for X
   - `libXp` - X11 X Print library

   If any one of the libraries is not present, the installer might revert to the console/text-based installer.
3. Check that there is sufficient free disk space.
4. If you are upgrading, stop any CICS Transaction Gateway application processes, product processes or services that are running, for example the Gateway daemon, the Client daemon and the Configuration Tool.
5. 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.
6. By default, the installer uses the `/tmp` directory for temporary data, or `/root` if there is insufficient space in `/tmp`. You can override this behavior by setting the *IATEMPDIR* environment variable to the required temporary directory before launching the installer.

# Chapter 20. File path terminology

Generic file paths are used to describe the location of files used by CICS Transaction Gateway.

## Installation directory

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

## Windows

You can choose the location for the product files during the installation of CICS Transaction Gateway on Windows, by default:

<install_path>

is:

- %Program Files%/IBM/CICS Transaction Gateway on 32-bit Windows systems.
- %Program Files(x86)%/IBM/CICS Transaction Gateway on 64-bit Windows systems.

## Common application data on Windows

On Windows the term <product_data_path> is the directory used by the CICS Transaction Gateway for common application data.

The <product_data_path> is a fully qualified directory path, the definition of <product_data_path> appends the Windows common application data directory with the subdirectory IBM/CICS Transaction Gateway.

When the Windows common application data directory is:

%ProgramData%

the <product_data_path> is:

%ProgramData%/IBM/CICS Transaction Gateway

The installation process creates environment variable *CTG_DATA_PATH* set with the value of <product_data_path>.

The Windows common application data directory is a hidden folder.

## UNIX

For new installations of CICS Transaction Gateway on UNIX:

<install_path> is /opt/IBM/cicstg

## Linux

For new installations of CICS Transaction Gateway on Linux:

<install_path> is /opt/ibm/cicstg

## Java home directory

The term <java_path> is used in file paths to represent the location of the Java home directory. For example /usr/lpp/java/J7.0.

## Software Developer Kit

The <SDK_path> is used to denote where the SDK has been installed.

# Chapter 21. Installing CICS Transaction Gateway on Windows

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.

Before installing the CICS Transaction Gateway, first follow the steps for Chapter 19, "Preparing to install CICS Transaction Gateway," on page 55.

## 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/Windows` directory.
3. Double-click on the `installer.exe`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 the following command: *d*`:/ctg/Windows/installer -i console` where *d*`:` is the DVD or network drive that contains the installation files.
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 called `installer.properties` in the specified directory or, if no directory is specified, in the current working directory. To generate a response file issue the command: **`installer -r <path_to_generate_response_file>`**.
2. Make the response file and the installation image available to the computer on which you are installing CICS Transaction Gateway. Ensure that the response file contains the property `LICENSE_ACCEPTED=TRUE`, to indicate that you accept the license agreement.
3. Issue the following command:

```
installer.exe -i silent -f C:/temp/installer.properties
    >C:/temp/installer.log 2>&1
```

An automatic restart is normal for an unattended upgrade installation.

### Unattended installation without a response file

To perform an unattended install without using a response file:

```
installer.exe -i silent -DLICENSE_ACCEPTED=true >C:/temp/installer.log 2>&1
```

An automatic restart is normal for an unattended upgrade installation.

## Location of the installation logs on Windows

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

Use the log files to diagnose any problems that cause an installation to fail. Any existing log files are overwritten by a new installation.

The following log files are created on Windows after a completed installation:
- <install_path>\installlogs\cicstgInstall.log
- <install_path>\installlogs\cicstgMessages.log
- <install_path>\installlogs\cicstgSymlinks.log
- <install_path>\installlogs\cicstgCRuntimeLibsInstall32.log
- <install_path>\installlogs\cicstgCRuntimeLibsInstall64.log (64-bit operating systems only)
- <install_path>\installlogs\cicstgFind.log

If the installation fails or the installation is canceled, the following log files are created on Windows:
- %TMP%\cicstgInstall.log
- %TMP%\cicstgFind.log

If an existing CICS Transaction Gateway could not be removed automatically during the upgrade the following log might also be created:
- %TMP%\cicstgUninstall.log

# Chapter 22. Installing CICS Transaction Gateway on UNIX and Linux

You can perform an install of CICS Transaction Gateway either using a GUI interface or a command prompt.

**Note:** Installation on a UNIX based platform may display the error message `Invalid unzip command found`. This error message can be ignored.

Before installing CICS Transaction Gateway, first follow the steps for Chapter 19, "Preparing to install CICS Transaction Gateway," on page 55.

## 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 disc into the disc drive and mount the DVD disc onto the mount point directory.
2. Navigate to the`/DVD/ctg/platform` directory, where *`DVD`* is the mount point directory and *`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 the DVD disc into the disc drive and mount the DVD disc onto the mount point directory.
2. Issue a command like the following:

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

   where *`DVD`* is the mount point directory 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 called `installer.properties` in the specified directory or, if no directory is

specified, in the current working directory. To generate a response file issue the command: **installer -r <path_to_generate_response_file>**.

2. Make the response file and the installation image available to the computer on which you are installing CICS Transaction Gateway. Ensure that the response file contains the property `LICENSE_ACCEPTED=TRUE`, to indicate that you accept the license agreement.
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
```

## Location of the installation logs on Unix and Linux

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

Use the log files to diagnose any problems that cause an installation to fail. Any existing log files are overwritten by a new installation.

The following log files are created on Unix and Linux after a completed installation:

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

If the installation fails or the installation is canceled, the following log files are created on Unix and Linux:

- /tmp/cicstgInstall.log
- /tmp/cicstgFind.log

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

- /tmp/cicstgUninstall.log

# Chapter 23. Installing a JVM

The CICS Transaction Gateway requires a supported version of Java. A JVM is installed with the CICS Transaction Gateway installation.

The default behaviour of CICS Transaction Gateway is to use the JVM installed with the product or you can configure a supported JVM that has been installed independently of CICS Transaction Gateway. See Specifying the JVM to configure the JVM to use.

The DVD and product download files for CICS Transaction Gateway for Multiplatforms provide JVM installers for use with remote Java applications.

The JVMs shipped with CICS Transaction Gateway may become out of date and potentially vulnerable. You should regularly check for later service releases of Java which might be available for download from FixCentral and should be used where available to ensure reliability, security, and compatibility with the latest features. For more information, see Chapter 26, “Installing and uninstalling fix packs,” on page 69.

# Chapter 24. Uninstalling CICS Transaction Gateway on Windows

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 uses the same method as the installation procedure.

## Uninstalling from the graphical user interface

To uninstall from the graphical user interface, start the uninstaller from the Windows Control Panel:

1. Click **Start**>**Settings**>**Control Panel**>**Add/Remove Programs**.
2. Select CICS Transaction Gateway in the list of programs and then click **Change/Remove**.

Alternatively, you can also launch the graphical user interface from the command line:

```
ctguninst -i gui
```

## Uninstalling from the command line

To uninstall from the command line:

```
ctguninst -i console
```

## Unattended uninstall using a response file

An unattended uninstall normally restarts the system.

To perform an unattended uninstall using a response file:

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

   ```
   ctguninst -i silent -f C:/temp/responsefile.txt
   ```

## Unattended uninstall without a response file

An unattended uninstall normally restarts the system.

To perform an unattended uninstall without a response file (where the default behaviour will preserve the configuration, log, and trace files:):

```
ctguninst -i silent
```

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

```
ctguninst -i silent -DPRESERVE_CONFIG_2=1
```

# Chapter 25. Uninstalling CICS Transaction Gateway on UNIX and Linux

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 uses the same method 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 console:

```
ctguninst -i console
```

**Unattended uninstall using a response file**

To perform an unattended uninstall 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
   ```

**Unattended uninstall without a response file**

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

```
ctguninst -i silent
```

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

```
ctguninst -i silent -DPRESERVE_CONFIG_2=1
```

# Chapter 26. Installing and uninstalling fix packs

Preparing a fix pack for installation or removal.

### Preparing to install or uninstall a fix pack

The fix pack utility updates an existing installation of CICS Transaction Gateway. Fix packs are available online at: http://www.ibm.com/support/fixcentral/.

The fix pack downloaded from Fix Central varies in size. The CICS Transaction Gateway for Multiplatforms fix pack is approximately 40 MB in size, the CICS Transaction Gateway Desktop Edition fix pack is approximately 25 MB in size. You need to unpack the downloaded fix pack file.

The fix pack utility runs in console mode and must be run by a user with a suitable level of authority. On Windows platforms, log on with administrator privileges; on UNIX and Linux platforms log on as root.

Before installing or uninstalling the fix pack, shut down any local CICS TG applications, the Gateway daemon, and then the Client daemon. Make sure that the Client daemon process **cclclnt** has ended.

On IBM AIX systems after the Client daemon has stopped, issue the command:
`/usr/sbin/slibclean`

### JVM packages

Three downloadable JVM packages are available:

- for HP platform
- for Oracle Solaris platform
- for IBM JVMs on other platforms: Windows, IBM AIX, Linux on POWER, Linux for Intel, and Linux for IBM z Systems

The actual releases available for a particular platform might vary, depending on the release schedule from the JTC.

## Installing a fix pack

Follow these steps to install a fix pack.

### Procedure

1. Change to the directory where the fix pack has been unpacked and issue the command: **./ctgfix**. The fix pack utility will prompt for confirmation before installing.
2. Enter 'y' to proceed.
3. Optional: To perform an unattended install, issue the command: **./ctgfix -y**

### What to do next

During installation, the fix pack utility creates a log file named `<install_path>/ctgfix/ctgfix.log`. The log file contains details of each file that is

updated by the fix pack, and details of any errors. Further information about error messages is available in text file `ctgfix.hlp` located where the fix pack was unpacked.

The fix pack utility updates Client daemon log messages to use US English. If a language other than US English is being used for Client daemon logging the **ctgmsgs** command must be run to change to the preferred language. For more details on running the **ctgmsgs** command, see Changing the system locale.

After installing the fix pack, the CICS Transaction Gateway version can be checked using the command **cicscli -v**. The fix pack utility does not update the Windows registry or the IBM AIX Software Vital Product Data database. The version of CICS Transaction Gateway displayed by Windows and IBM AIX system utilities remains as the original installed version.

## Uninstalling a fix pack

Follow thse steps to uninstall a fix pack.

### Procedure

1. Change to the `<install_path>/ctgfix` directory and issue the command: **./ctgfix -u**. The fix pack utility will prompt for confirmation before uninstalling.
2. Enter 'y' to proceed.
3. Optional: To perform an unattended uninstall, issue the command: **./ctgfix -u -y**

### What to do next

During uninstall, the fix pack utility appends messages to the `ctgfix.log` file. The log file contains details of each file that is restored by the fix pack, and details of any errors. Further information about error messages is available in text file `<install_path>/ctgfix/ctgfix.hlp`. After uninstalling the fix pack, the CICS Transaction Gateway version can be checked by issuing the command: **cicscli -v**

# Chapter 27. Software Development Kit

The CICS TG Software Development Kit (SDK) contains components to develop CICS TG applications and user exits.

You can download the SDK package from https://developer.ibm.com/cics/2016/03/11/cics-tg-sdks/. The documentation on the download site contains installation instructions and information about which components can be redistributed with third party applications.

# Chapter 28. Using X Window System for GUI

The X Window System software can be used to provide a graphical user interface (GUI) for computers running UNIX and Linux operating systems.

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 displays 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 are displayed (followed by a colon and the display ID, which is normally 0.0). The application windows are then displayed on the *disp* system.

# Part 5. Upgrading

Use this information to plan and upgrade your environment and applications to work in CICS Transaction Gateway V9.2.

# Chapter 29. Upgrading from CICS Transaction Gateway Version 7 or later

Use this information when you plan to upgrade from CICS Transaction Gateway Version 7 or later to CICS Transaction Gateway V9.2.

The following table lists the changes in CICS Transaction Gateway which may affect the way you upgrade to CICS Transaction Gateway V9.2.

*Table 3. Upgrading cross reference*

| Feature or change | From V9.1 | From V9.0 | From V8.1 | From V8.0 | From V7.2 | From V7.1 | From V7.0 |
|---|---|---|---|---|---|---|---|
| **Configuring** | | | | | | | |
| "APPLID and APPLID qualifier length limit" on page 79 | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| "Supported characters in server names" on page 79 | | | | | ✓ | ✓ | ✓ |
| "Upgrading a statistics API port definition" on page 79 | | | | | ✓ | ✓ | |
| "Removal of TCP62 support" on page 80 | | | | | | | ✓ |
| "Removal of SNA Server parameter **LUALIASNAMES** on Windows" on page 80 | | | | | | | ✓ |
| "Removal of named pipe protocol support on Windows" on page 80 | | | | ✓ | ✓ | ✓ | ✓ |
| "IPIC server idle timeout default setting" on page 80 | | | ✓ | ✓ | ✓ | ✓ | |
| | | | ✓ | ✓ | ✓ | ✓ | ✓ |
| "Environment variable *CTG_JAVA* on UNIX and Linux" on page 81 | | | ✓ | ✓ | | | |
| **Operating** | | | | | | | |
| "Log file for the ctgd command" on page 82 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| "Changes to configuration files on UNIX and Linux" on page 82 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| "Changes to Gateway daemon log and trace files on UNIX and Linux" on page 82 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| "Improved configuration file checking" on page 82 | | | | | ✓ | ✓ | ✓ |
| "Client log file text encoding on Windows" on page 83 | | | ✓ | ✓ | ✓ | ✓ | ✓ |

*Table 3. Upgrading cross reference (continued)*

| Feature or change | From V9.1 | From V9.0 | From V8.1 | From V8.0 | From V7.2 | From V7.1 | From V7.0 |
|---|---|---|---|---|---|---|---|
| "Java Version 7 is required by the Gateway daemon" on page 83 | | | ✓ | ✓ | ✓ | ✓ | ✓ |
| "Java on UNIX and Linux" on page 83 | | | ✓ | ✓ | ✓ | ✓ | ✓ |
| "Changes to Windows services" on page 83<br>• "Location of configuration, log and trace files on Windows" on page 84<br>• "Removal of the **ctgstart** command on Windows" on page 84 | | | | | ✓ | ✓ | ✓ |
| "SSL keyring settings moved" on page 84 | | | ✓ | ✓ | ✓ | ✓ | ✓ |
| "TLS cipher suites" on page 84 | | | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Applications and user exits** | | | | | | | |
| Software Development Kit available only as separate download | ✓ | | | | | | |
| "Installed API documentation files moved to SDK" on page 85 | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| "Existing ECI V2 and ESI V2 applications on HP-UX" on page 85 | | | ✓ | ✓ | ✓ | | |
| "Assembly redirection for Microsoft NET Framework-based applications" on page 85 | ✓ | ✓ | ✓ | ✓ | | | |
| "Removed dependency on ECI V2 for Microsoft NET Framework-based applications" on page 86 | | | | ✓ | | | |
| "Message_Qualifier API removal" on page 87 | | | ✓ | ✓ | ✓ | ✓ | ✓ |
| "Using the JEE interfaces in nonmanaged mode" on page 87 | | | | ✓ | ✓ | ✓ | ✓ |
| "ctgredist archive replaced" on page 87 | | ✓ | ✓ | ✓ | ✓ | | |
| **Other** | | | | | | | |
| "Upgrading from CICS Universal Client" on page 87 | | | | | | ✓ | ✓ |

# Configuring

This section contains information about changes to the way you configure CICS Transaction Gateway V9.2.

Refer to the Upgrading cross reference table to find which of these tasks are applicable to your upgrade.

## APPLID and APPLID qualifier length limit

This information describes changes to the maximum length allowed for APPLIDs and APPLID qualifiers in client applications introduced in CICS Transaction Gateway Version 9.1.

In CICS Transaction gateway V9.0 and earlier, the 8 character limit for APPLIDs and APPLID qualifiers was documented but not enforced. This has been changed so that requests that specify APPLIDs or APPLID qualifiers that are longer than 8 characters are rejected and message CTG6675E or CTG6676E will be issued. Existing client applications containing APPLIDs or APPLID qualifiers that exceed the maximum length should be updated so that they are no longer than 8 characters.

## Supported characters in server names

This information details the supported characters in server names introduced in CICS Transaction Gateway Version 8.0.

Server names must now use characters from the supported character list to ensure that all CICS TG functions work correctly. Existing configuration files containing server names using unsupported characters can continue to be used as an aid to migration but might not work in all scenarios. Configuration files containing server names that use unsupported characters should be migrated as soon as possible.

For the list of supported characters, see the relevant page on configuring the server name for the required protocol:

- TCP server names
- SNA server names
- IPIC server names

.

## Upgrading a statistics API port definition

This information details the changes that you need to consider when upgrading a statistics API port definition from CICS Transaction Gateway V7.1

If you configured a statistics API port defined by a **statsport** parameter in the GATEWAY section of your configuration file, upgrade to using a full statistics API 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 statistics API protocol handler the remote monitoring applications can connect to the Gateway daemon. See Statistics API protocol settings for details on remote statistics API connections.

## Removal of TCP62 support

This information describes the TCP62 support that was removed in CICS Transaction Gateway Version 7.1.

The releases of CICS Transaction Gateway V7.0 and CICS Universal Client V7.0 are the last releases that contained TCP62 support for the IBM AnyNet® protocol communicating with remote CICS systems using SNA over TCP/IP protocol encapsulation. Accordingly, this capability is 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 IBM AnyNet support from IBM z/OS Communications Server, refer to the IBM z/OS and IBM z/OS statements of direction announcement, Software Announcement 203-266, dated October 7, 2003, and the IBM z/OS V1.7 preview announcement, Software Announcement 205-034, dated February 15, 2005.

## Removal of SNA Server parameter `LUALIASNAMES` on Windows

This information describes the **`LUALIASNAMES`** parameter that was removed in CICS Transaction Gateway Version 7.1.

On Windows LUALIASNAMES is replaced by LOCALLUALIAS and 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.

## Removal of named pipe protocol support on Windows

This information details the removal of named pipe protocol support on Windows introduced in CICS Transaction Gateway Version 7.1.

The named pipe protocol for Windows is no longer supported. To enable communication with local IBM TXSeries systems, replace all named pipe server definitions with TCP/IP server definitions.

## IPIC server idle timeout default setting

This information describes changes to the IPIC server idle timeout default setting that were introduced with CICS TG v9.0.

The default setting of the server idle timeout period for IPIC server connections (see “Server idle timeout” on page 121) has been changed to zero, so that the idle timeout period is disabled. Previously, IPIC server connections would be closed if idle for more than 60 minutes. This change affects local mode topologies, and also remote mode topologies which do not configure an IPIC server idle timeout.

## Environment variable *CTG_JAVA* on UNIX and Linux

The environment variable *CTG_JAVA* does not need to be defined for CICS Transaction Gateway Version 9.0 and later.

The default behaviour of commands **ctgadmin**, **ctgcfg**, **ctgd**, and **ctgstart** has changed to use the Java installed with CICS Transaction Gateway.

If the variable *CTG_JAVA* is defined you must remove it from the CICS TG environment or you must redefine *CTG_JAVA* to use a supported version of Java.

## Removal of Gateway daemon resource parameters

This information details changes to the Gateway daemon resource parameters that was introduced in CICS Transaction Gateway Version 7.2.

The **ecigenericreplies**, **msgqualvalidation**, and **uowvalidation** parameters are no longer supported. This means that ECI_GET_REPLY and ECI_GET_REPLY_WAIT call types are no longer available in remote mode and message qualifiers and LUW tokens are validated so that they can be used only on the JavaGateway connection to that they were allocated.

If one or more of these parameters are specified in the configuration file, the Gateway daemon ignores the value and startup continues. If you use the Configuration Tool to edit a configuration file that contains any of these parameters, the Configuration Tool removes all instances of the deprecated parameters when the configuration file is updated. If you do not use the Configuration Tool to edit your configuration files, you must manually remove the deprecated parameters from the configuration file.

If any of your applications rely on the deprecated parameters, the following situations occur:

- Java Client applications running in remote mode that use the ECI_GET_REPLY or ECI_GET_REPLY_WAIT call types receive the ECI_ERR_INVALID_CALL_TYPE error when they flow the request to the Gateway daemon. Java Client applications running in local mode can use ECI_GET_REPLY and ECI_GET_REPLY_WAIT call types.
- Any remote mode Java Client application that relies on the **msgqualvalidation** parameter being turned off receives the ECI_ERR_NO_REPLY error when trying to use the message qualifier on a connection that did not start the asynchronous pieces of work.
- Any application that relies on the **uowvalidation** parameter being turned off receives the ECI_ERR_LUW_TOKEN error when trying to use the LUW token on a connection that did not start the LUW.

ECI V2 asynchronous call support is unaffected by the deprecation of these parameters.

# Operating

This section contains information on changes to the way you operate CICS Transaction Gateway V9.2.

Refer to the Upgrading cross reference table to find which of these tasks are applicable to your upgrade.

**Note:** The operating systems supported by the latest version of CICS Transaction Gateway will change as new operating systems are released. Before upgrading to CICS Transaction Gateway V9.2 check the operating system is still supported, refer to "Operating systems" on page 27.

## Log file for the ctgd command

The **ctgd start** command creates a log file automatically.

The **ctgd** command on UNIX and Linux was changed in CICS Transaction Gateway V9.2 so that the **ctgd start** command creates a log file called `/var/cicscli/ctgd.log` automatically. For more information about using the **ctgd** command see: "Running the Gateway daemon as a background process" on page 333.

## Changes to configuration files on UNIX and Linux

This information describes the configuration changes introduced in CICS Transaction Gateway V9.2.

The default location of the configuration files has moved from `<install_path>/bin` to `/var/cicscli`. The installer automatically moves existing configuration files from `<install_path>/bin` to `/var/cicscli` during an upgrade, provided that files of the same name are not already in `/var/cicscli`. The configuration files include: `ctg.ini`, `cicscol.ini`, `cicskey.ini`, and `ctgd.conf`.

For more information about `<install_path>` see Chapter 20, "File path terminology," on page 57

The **ctgd** command on UNIX and Linux was changed in CICS Transaction Gateway V9.2 so that it is not necessary to define a `ctgd.conf` configuration file. If the *CTGDCONF* variable is not defined and the `var/cicscli/ctgd.conf` file does not exist, the **ctgd** command will use the value set in the **adminport** parameter in the `ctg.ini` configuration file. If this parameter is not set, the default 2810 is used. All other parameters will be set to default values. For more information, see the "ctgd command reference" on page 364.

## Changes to Gateway daemon log and trace files on UNIX and Linux

This information describes the changes to the location of log and trace files introduced in CICS Transaction Gateway V9.2.

On UNIX and Linux, the default log destination has been changed from `console` to `file`, and the default location of the Gateway daemon log and trace files has changed from `<install_path>/bin` to `/var/cicscli`. The files affected are: Gateway information and error logs, Gateway trace, and JNI trace. For more information see "Trace options" on page 345 and "Gateway daemon logging" on page 161.

The Configuration Tool has been changed to use the new default logging type and file locations.

## Improved configuration file checking

This information describes the enhanced checking of configuration files that you need to consider when upgrading from CICS Transaction Gateway Version 7.2 or earlier.

CICS Transaction Gateway configurations that worked with previous releases might not work after you upgrade. Configuration checking is enhanced to ensure that the values used by CICS Transaction Gateway are the intended ones. Protocol handlers are not started unless explicitly configured.

## Client log file text encoding on Windows

This information details changes to the Client log file text encoding on Windows which was introduced in CICS Transaction Gateway Version 9.0.

The Client daemon information and error log files are now written in the UTF-8 text encoding on Windows. When you use existing log files that were created with an earlier version of CICS TG, new log messages might appear incorrectly due to the change of text encoding. To avoid this, rename or remove existing Client log files before starting CICS Transaction Gateway, or update your CICS Transaction Gateway configuration to use new Client daemon log files. For more information, see "Client daemon logging" on page 184.

## Java Version 7 is required by the Gateway daemon

This information details the Java runtime environment that is required by CICS Transaction Gateway V9.2.

On IBM AIX, Linux and Windows operating systems, CICS Transaction Gateway requires the Java 7.1 runtime environment. On HP-UX and Solaris operating systems CICS Transaction Gateway requires the Java 7.0 runtime environment. For more information, see "Java runtime environments for core components" on page 28. A supported Java runtime environment is installed with the product.

For more information on the default Java location and defining *CTG_JAVA* on UNIX and Linux, see "Java on UNIX and Linux."

To specify an alternate JVM see, "Specifying the JVM" on page 95

## Java on UNIX and Linux

This information describes the changes to invoking Java which was introduced in CICS Transaction Gateway Version 9.0 on UNIX and Linux systems.

The default behaviour of commands **ctgadmin**, **ctgcfg**, **ctgd**, and **ctgstart** has changed to use the Java installed with CICS Transaction Gateway. Prior to CICS Transaction Gateway Version 7.2 these commands used the Java found on the *PATH*.

CICS Transaction Gateway Version 7.2 introduced the environment variable *CTG_JAVA* to define which Java executable is invoked. If **CTG_JAVA** is defined you must remove it from the CICS TG environment or you must redefine *CTG_JAVA* to a supported version of Java.

To specify an alternate JVM see, "Specifying the JVM" on page 95

## Changes to Windows services

This information describes the changes to Windows services introduced in CICS Transaction Gateway on Windows Version 8.0.

The CICS Transaction Gateway now runs as a single service on all supported Windows operating systems. Messages associated with running the CICS Transaction Gateway as a service are written to the Application section of the Windows Event log.

## Location of configuration, log and trace files on Windows

This information describes the changes to the location of configuration, log and trace files introduced in CICS Transaction Gateway Version 8.0.

The default location of configuration, log and trace files is changed from `<install_path>/bin` to `<product_data_path>`. The installer automatically moves configuration files from `<install_path>/bin` to `<product_data_path>`. For more information see Chapter 20, “File path terminology,” on page 57

## Removal of the `ctgstart` command on Windows

This information describes the **ctgstart** command which was removed in CICS Transaction Gateway Version 8.0.

The Gateway daemon always runs as a Windows service and it is no longer possible to operate the Gateway daemon in console mode. The **ctgstart** command is removed from the product and the **ctgadmin** command is extended to support starting the Gateway daemon.

## SSL keyring settings moved

This information details changes to the SSL keyring settings introduced in CICS Transaction Gateway Version 9.0.

The SSL key ring settings are now product wide; they have been moved from the SSL protocol handler in the GATEWAY section to the PRODUCT section of the configuration file. The same SSL key ring settings are used for both SSL protocol handler and IPIC server SSL connection definitions. The SSL key ring parameters must be defined in the PRODUCT section in order to use IPIC over SSL. The definition of the SSL key ring parameters in the GATEWAY section is supported, if not using IPIC over SSL, for migration purposes. The SSL key ring settings are: **keyring**, **keyringpw**, and **keyringpwscrambled**.

## TLS cipher suites

This information details changes to the SSL keyring settings introduced in CICS Transaction Gateway Version 9.0.

Cipher suites entered as TLS are no longer converted to SSL when CICS Transaction Gateway starts.

# Applications and user exits

This section contains information on changes that you may need to make to your applications when you upgrade to CICS Transaction Gateway V9.2.

Refer to the Upgrading cross reference table to find which of these tasks are applicable to your upgrade.

## Installed API documentation files moved to SDK

In CICS Transaction Gateway Version 9.1, API documentation files were moved to the Software Development Kit (SDK).

Use the Software Development Kit when developing on a different system to where the CICS TG product has been installed or for re-distributable files. API documentation files were also moved to the SDK.

**Note:** In CICS Transaction Gateway Version 9.2, the Software Development Kit is no longer shipped as part of the product, and must be downloaded separately.

## Existing ECI V2 and ESI V2 applications on HP-UX

This information details changes to the existing ECI V2 and ESI V2 applications on HP-UX that you need to consider when upgrading.

Existing single threaded ECI V2 and ESI V2 applications on HP-UX that are not explicitly linked with the `libpthread` shared library must be rebuilt to link with `libpthread`. Applications that are not linked with `libpthread` might fail to establish a connection when connecting to the Gateway daemon with the error `CTG_ERR_CONNECTFAILED`.

For more information, see the *CICS Transaction Gateway for IBM z/OS: Developing Applications*.

## Assembly redirection for Microsoft NET Framework-based applications

You can use assembly redirection to configure your applications to use an updated version of the CICS Transaction Gateway .NET assembly (`IBM.CTG.Client.dll`) without recompiling the applications.

This change will affect you if you are upgrading from V9.1, V9.0, V8.1 or V8.0.

Assembly redirection is possible either at the application level or the machine level. For an assembly redirection at the machine level using the Global Assembly Cache (GAC), you must install the upgraded assembly, and the publisher policy assembly into the GAC. The assembly is provided with the product installation in the `<install_path>/lib` directory; and with the SDK package in the `<SDK_path>/api/dotnet/runtime` directory. Publisher policy assemblies are provided for upgrading .NET applications. The policy files are in the `<install_path>/lib/policy` directory for the product installation and in `<SDK_path>/api/dotnet/runtime/policy` for the SDK package. For more information see the .NET Microsoft documentation.

For an application level or machine level upgrade, you can optionally add the following code to the application configuration file or machine configuration file:

```
<configuration>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="IBM.CTG.Client"
          publicKeyToken="4f7d883847d47abe"
          culture="neutral" />
        <bindingRedirect oldVersion="x.x.0.0-x.x.0.9" newVersion="9.2.0.0" />
```

```
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

where x.x is the version and release of the CICS TG version you are upgrading from. For example, if you are upgrading from version 8.1 the value should appear as: oldVersion="8.1.0.0-8.1.0.9"

## Removed dependency on ECI V2 for Microsoft NET Framework-based applications

This change will affect you if you upgrade from CICS TG V8.0.

The CICS Transaction Gateway API for .NET Framework has been upgraded to support Microsoft 32-bit and 64-bit Windows architectures from a single assembly (IBM.CTG.Client.dll).

The upgraded assembly is included in the SDK package in `<SDK_path>/api/dotnet/runtime`, or in `<install_path>/lib` on a Windows machine with CICS Transaction Gateway installed. The upgraded CICS Transaction Gateway API for .NET Framework does not depend on ECI V2 and the upgraded `IBM.CTG.Client.dll` does not need `ctgclient.dll` referenced in the PATH.

The behavior of the Gateway trace, ECI timeout, and ECI request extend mode functions is different when compared with earlier versions of CICS Transaction Gateway:

- Trace from the CICS Transaction Gateway API for .NET Framework is written using System.Diagnostics.Trace; the traceFile attribute is no longer used by the CtgTrace switch or corresponding `IBM.CTG.Trace.SetTraceFile(string fileName)` method. The switch and the method can still be accessed by applications but do not provide any function.
- The upgraded CICS Transaction Gateway API for .NET Framework does not support enabling trace with environment variables.
- If the value of the `IBM.CTG.EciRequest.Timeout` property is negative an `ArgumentOutOfRangeException` occurs.
- If the value of the `IBM.CTG.EciRequest.ExtendMode` property is not a defined `IBM.CTG.EciExtendMode` enumeration an `ArgumentOutOfRangeException` occurs.

Applications based on Microsoft .NET Framework 3.5 do not require modification; they can be left with their dependency on `ctgclient.dll`. Alternatively, applications can use assembly redirection to access the upgraded assembly. Assembly redirection is possible either at the application level or the machine level. For an assembly redirection at the machine level using the Global Assembly Cache (GAC), you must install the upgraded assembly, and the publisher policy assembly, into the GAC.

Refer to “Assembly redirection for Microsoft NET Framework-based applications” on page 85

If you are using an application that targets Microsoft .NET Framework 4.0, to use the upgraded assembly you must recompile the application.

## Message_Qualifier API removal

This information details the removal of the Message_Qualifier API that you need to consider when upgrading from CICS TG V8.1 or earlier.

The deprecated field Message_Qualifier has been removed from the Java API. Applications that used this field will need to use the `getMessageQualifier()` and `setMessageQualifier()` methods instead.

Java client applications using the deprecated field can still connect to CICS TG V9.2 when run in remote mode using a CICS TG V8.1 or earlier `ctgclient.jar` file.

## Using the JEE interfaces in nonmanaged mode

In CICS Transaction Gateway Version 8.1 the JAR file `cicsj2ee.jar` file was renamed to `cicsjee.jar`.

CLASSPATH definitions that include `cicsj2ee.jar` should be changed to use `cicsjee.jar`.

## ctgredist archive replaced

In CICS Transaction Gateway Version 9.1 the ctgredist archive file was replaced by the Software Development Kit.

Use this new archive when developing on a different system to where the CICS TG product has been installed or for redistributable files.

For more information about the Software Development Kit, see Chapter 27, "Software Development Kit," on page 71*Software Development Kit*

# Upgrading from CICS Universal Client

This information details changes that you need to consider when upgrading to CICS Transaction Gateway 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 other upgrading topics 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.

# Chapter 30. Upgrading from Version 6 and earlier

For product changes from Version 6 and earlier, to Version 7 and later, refer to the appropriate documentation.

**Note:** When you prepare to upgrade from an unsupported release of CICS Transaction Gateway to CICS Transaction Gateway V9.2, you should check the upgrading documentation for an interim release, such as V7.1 for any special requirements. Alternatively, you can install CICS Transaction Gateway V9.2 as a new installation.

For more information, see Documentation for older versions of CICS products.

# Part 6. 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.

**Related information**:

"Running the Gateway daemon as a background process" on page 333
On UNIX and Linux you can run the Gateway daemon as a background process with preset options defined in `ctg.ini` or you can specify override options using the **ctgd** command.

Chapter 62, "Starting CICS Transaction Gateway on Windows," on page 337
CICS Transaction Gateway is installed and runs as a Windows service, so it can be started and stopped in the same way as any Windows service.

# Chapter 31. Creating a configuration file

Use the Configuration Tool to create a CICS Transaction Gateway configuration file.

To start the Configuration Tool, enter the **ctgcfg** command at a command prompt. Alternatively, on Windows, click the shortcut on the **Start** menu. To use the Configuration Tool remotely on UNIX and Linux, export the display using the commands described in Using X-Window System from a remote system.

Configuration settings are stored by default in the `ctg.ini` file, in the `/var/cicscli` directory on UNIX and Linux, and in the `<product_data_path>` directory on Windows. You can specify a different location and optionally change the name of the configuration file by setting the *CICSCLI* environment variable. For more information, see "Specifying the configuration file" on page 95. If this variable is set, the Configuration Tool loads the file referenced by *CICSCLI,* when it starts, otherwise it attempts to load `ctg.ini` from the default location. If the Configuration Tool fails to find a configuration file, it creates a new one. This file must be edited and saved before use.

The Configuration Tool allows parameter values to be entered in mixed-case and also writes the values as mixed-case to the configuration file. Some values are changed to uppercase at run time.

The CICS Transaction Gateway configuration file is processed during CICS Transaction Gateway initialization. If changes are made to the configuration file whilst CICS Transaction Gateway is running, the changes will have no effect until CICS Transaction Gateway is restarted.

## Running the Configuration Tool for a different operating system

Proceed as follows:

1. Install the product on the workstation where you want to run the Configuration Tool.
2. To edit the configuration, copy the configuration file (`ctg.ini`) to the `<product_data_path>` directory 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.

# Chapter 32. Configuring the system environment

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

## Configuring the Windows service

You are recommended to start the IBM CICS Transaction Gateway service automatically at Windows startup, after the CICS Transaction Gateway has been correctly configured.

You must have Administrator authority to configure service startup.

1. Select the service in the services dialog.
2. From the menu bar select **Action –> Properties**.
3. Change the **Startup type** to *Automatic*.
4. Select the **Log On** tab.
5. Make sure that the **Local System Account** radio button is selected.

To permanently configure startup override parameters see, "Setting CICS Transaction Gateway startup override options" on page 337.

## Specifying the configuration file

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

**Windows**

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

```
<product_data_path>/ctg.ini
```

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

```
set CICSCLI=D:/shared/ctg.ini
```

**UNIX and Linux**

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

```
/var/cicscli/ctg.ini
```

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

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

## Specifying the JVM

You can override the default JVM used by CICS Transaction Gateway.

By default, CICS Transaction Gateway uses the JVM which is installed in `<install_path>/jvm170`. You can override the default java as described below.

### Windows

To set the JVM used by CICS Transaction Gateway, use the **ctgjava** command from an administrator level command prompt.

Use the **ctgjava** command to specify which JVM CICS Transaction Gateway uses:

```
ctgjava -s=jvmlocation
```

where *jvmlocation* is the fully qualified path name of the Java executable file.

If the path name contains spaces, enclose it in quotation marks. For example:

```
ctgjava -s="C:/Program Files/IBM/CICS Transaction Gateway/jvm170/bin/java.exe"
```

To check which JVM CICS Transaction Gateway uses:

```
ctgjava -v
```

To have a JVM selected automatically:

```
ctgjava -a
```

### UNIX and Linux

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:

- IBM AIX, Solaris and HP-UX

  ```
  export CTG_JAVA=/opt/IBM/cicstg/jvm170/bin/java
  ```

- Linux

  ```
  export CTG_JAVA=/opt/ibm/cicstg/jvm170/bin/java
  ```

## Changing the system locale

The system locale is automatically recognized by the Gateway daemon, the configuration tool and the **ctgadmin** command. Other components of the product use language files that must be changed when the locale is changed.

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

1. Stop the CICS Transaction Gateway and the Client daemon.
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:

   On UNIX and Linux platforms:

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

   On Windows platforms:

   ```
   ctgmsgs XX [-silent]
   ```

   where *XX* is the two character language identifier.

```
XX  Language
    --------------------
en  US English
fr  French
de  German
it  Italian
es  Spanish
tr  Turkish
ja  Japanese
ko  Korean
zh  Simplified Chinese
```

*Figure 5. The supported languages*

On UNIX and Linux systems enter the **ctgmsgs** command with no parameters to list the supplied code sets.

4. Restart the CICS Transaction Gateway.

## Using an alternative code set on IBM 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
```

Complete 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 HP-UX, Solaris and Linux

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 message queues on UNIX and Linux

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

These are the suggested settings for message queues on HP-UX.

### HP-UX 11i V3

```
msgmbs 8 Total Message Space for Sys V IPC Message Queues (MB)
msgmnb 65535 Maximum size of a message queue (bytes)
```

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

1. Type **smh** at the command prompt.
2. Click **Kernel Configuration** > **Tunables**.

   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. Click **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, SMH displays a window explaining the error.

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

   SMH displays a window asking for confirmation; click **Yes**.

SMH 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.

### HP-UX 11i V2

```
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. Click **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. Click **Actions** > **Modify configurable parameter**.
5. Enter the new value for the parameter in the `Formula/Value` field, and then click **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, click **Actions** > **Process New Kernel**.

   SAM displays a window asking for confirmation. Click **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

These are the suggested settings to place in the configuration 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 are the suggested settings for message queues on Solaris.

You can set these values by changing the entries in the `/etc/system` file.

Suggested settings for message queues on Solaris.

| Setting | Description |
| --- | --- |
| `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. |
| `set msgsys:msginfo_msgseg = 16384` | Number of `msginfo_msgssz` segments the system uses as a pool for available message memory.<br><br>Total memory available for messages is `msginfo_msgseg` * `msginfo_msgssz`.<br><br>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.<br><br>A low value can adversely affect system performance, or cause the client to freeze. You should 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. |
| | |

# Chapter 33. 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 Chapter 37, "Configuring CICS server connections," on page 111.
2. If you are using TCP/IP or SNA connections, configure the Client daemon settings, as described in Chapter 39, "Configuring Client daemon settings," on page 179.
3. If you are using a Java client, it must be able to find `ctgjni.dll` on Windows or `libctgjni.so` on UNIX and Linux. Make one of the following configuration changes:
   - Add **`-Djava.library.path=<install_path>/bin`** to the Java client start parameters.
   - On Windows, set the *PATH* environment variable to include the `<install_path>\bin` directory.
   - On UNIX and Linux, set `LD_LIBRARY_PATH` to include the `<install_path>/bin` directory.
4. If you are using a JEE application server, deploy the resource adapter that you require, as described in Chapter 93, "Deploying a CICS resource adapter," on page 451.

# Chapter 34. Configuring a remote mode topology

In a remote topology, the Gateway daemon handles connections to remote client applications.

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

1. Configure Gateway daemon settings as described in Configuring Gateway daemon settings This includes configuring a protocol handler to handle remote client connections.
2. If you are using TCP/IP or SNA connections, configure the Client daemon settings, as described in Configuring Client daemon settings.
3. Configure the CICS server connections, as described in Configuring CICS server connections.
4. Configure other features that you require such as high availability or monitoring, for example.

## Setting Gateway daemon JVM options

Startup options are used to set the properties of the Gateway daemon JVM, such as heap size and Java properties.

### UNIX and Linux

When starting the Gateway daemon in console mode, JVM options can be specified in the parameters passed to the **ctgstart** command. Each JVM option must be prefixed with `-j`. For more information see "Starting the Gateway daemon in console mode" on page 333. For example, to set the maximum JVM heap size to 512MB:

```
ctgstart -j-Xmx512M
```

You can query the maximum JVM heap size with the following command:

```
ctgadmin -a stats -gs SE_SHEAPMAX
```

If you have the CICS TG plug-in installed in Explorer, you can also see the value in the `MaxHeap` column.

When running the Gateway daemon as a background process, JVM options can be specified in the *CTGD_PARAMS* variable in the `ctgd` configuration file `ctgd.conf`. Each JVM option must be prefixed with `-j`. For more information see "Running the Gateway daemon as a background process" on page 333.

For example, to set the maximum JVM heap size to 512MB:

```
CTGD_PARAMS=-j-Xmx512M
```

### Windows

JVM options can be specified in the parameters set using the **ctgservice** command. Each JVM option must be prefixed with `-j`, and all options must be prefixed with `-A` when using the **ctgservice** command. For more information see "ctgservice command reference" on page 365.

For example, to set the maximum JVM heap size to 512MB:

```
ctgservice -R -A-j-Xmx512M
```

You can query the maximum JVM heap size with the following command:

```
ctgadmin -a stats -gs SE_SHEAPMAX
```

If you have the CICS TG plug-in installed in Explorer, you can also see the value in the `MaxHeap` column.

# Chapter 35. Configuring for application testing

The CICS Intercept plug-in can intercept ECI, EPI and ESI requests for any remote mode API.

The CICS Intercept plug-in is configured using the **cicsintercept** parameter in the "GATEWAY section of the configuration file" on page 240.

```
cicsintercept=org.example.intercept.ClassName
```

If the plug-in fails to load then the Gateway daemon will fail to start and an error message is written to the log. If the plug-in is loaded successfully then a message is written to the information log.

**Note:** Statistics and Monitoring exits are not guaranteed to be accurate when a CICS Intercept plug-in is enabled. If a CICS Intercept plug-in returns a request rather than letting it continue, then the following needs to be considered:

- A configured CICS Request exit will not be called.
- Any configured Request Monitoring exits will only be called for RequestEntry and ResponseExit.
- GD, CS and CSx statistics will not be updated.

## CICS Intercept plug-in

The **cicsintercept** parameter specifies the class used to intercept ECI, EPI, and ESI requests for any remote mode API.

**cicsintercept=**<class>

**Description**

Set the value to a fully qualified class name that implements the com.ibm.ctg.intercept.CicsIntercept interface. The Gateway daemon supports only a single CICS Intercept plug-in and the class must be on the class path of the Gateway daemon. This parameter is in the GATEWAY section of the configuration file. For more information see Chapter 35, "Configuring for application testing."

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**

There is no default value.

**Configuration Tool**

In the Configuration Tool, you can set the value of **cicsintercept** in the **CICS Intercept plug-in** field in the **Resources** tab of the **Gateway daemon settings** page.

# Chapter 36. Configuring 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 IBM CICSPlex®, and for subsequent task correlation in CICS.

Gateway identification has implications for the following:

- IPIC connections to identify the Gateway daemon to CICS
- SNA and TCP/IP connections to identify the Gateway daemon to CICS
- Request monitoring data

### 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 109 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” on page 108.

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.

### Request monitoring data

The APPLID is available to the request monitoring exits and also transaction tracking across a IBM CICSPlex. For more information, see Monitoring and statistics.

## Gateway APPLID

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

**applid=<name>**

**Description**

Set a value of up to eight characters. The value must be unique within the IBM CICSPlex. There is no restriction on the characters that can be used. However, to ensure that the **applid** parameter is valid for all scenarios, use characters in the range A through Z, and 0 through 9. For connections to CICS servers, the **applid** parameter value is converted to uppercase. If you do not set a value, the system automatically generates a unique value. The combination of the **applid** and **applidqualifer** parameters identifies CICS Transaction Gateway to the CICS system to which it connects.

This parameter is in the PRODUCT section of the configuration file.

**Default value**
There is no default value for this parameter.

**Configuration Tool**
In the Configuration Tool, you can set the value of the **applid** parameter in the **APPLID** field on the **CICS Transaction Gateway** page.

## Gateway APPLID qualifier

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

**applidqualifier=<name>**

**Description**
Set a value of up to eight characters. There is no restriction on the characters that can be used. However, to ensure that the **applidqualifier** parameter is valid for all scenarios, use characters in the range A through Z, and 0 through 9. The combination of the **applid** and **applidqualifier** parameters identifies CICS Transaction Gateway to the CICS system to which it connects.

**Default value**
If the **applid** parameter is specified in the configuration file but the **applidqualifier** parameter is not specified, the system uses the default, 9UNKNOWN, value for the **applidqualifier** parameter. 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, in CICS, and in statistics. Having a default provides a reference value to make problem diagnosis simpler.

**Configuration Tool**
In the Configuration Tool, you can set the value of **applidqualifier** parameter in the **APPLID qualifier** field in the **CICS Transaction Gateway** page.

## Client APPLID and APPLID qualifier

Set the APPLID and APPLID qualifier for Client applications to enable transaction tracking.

There is no restriction on the characters that can be used for APPLID and APPLID qualifier, however, to ensure that the APPLID and APPLID qualifier are valid for all scenarios, use characters in the range A through Z, and 0 through 9.

### Java clients

Set the fully qualified APPLID for Java clients in one of the following ways:

- For JEE applications, set the Applid and ApplidQualifier properties in the resource adapter custom properties to specify the fully qualified APPLID.
- For Java client applications, set the APPLID and APPLID qualifier in the Properties object for the JavaGateway class as follows:

  **APPLID**
  CTG_APPLID

  **APPLID qualifier**
  CTG_APPLIDQUALIFIER

- Alternatively, for existing applications that you do not want to recompile, you can set JVM properties using the following system properties:

  **APPLID**
  CTG_APPLID

  **APPLID qualifier**
  CTG_APPLIDQUALIFIER

  For example:

  ```
  java -DCTG_APPLID=myapplid -DCTG_APPLIDQUALIFIER=applqual my.ctg.application
  ```

  If the APPLID or APPLID qualifier is set using a JVM property and using a Properties object passed to the JavaGateway, the JVM property value is used.

### ECI V2 clients

For more information see the *Programming Guide*.

### .NET Framework-based clients

For more information see GatewayConnection class.

## 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 108.

### 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 connection's 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.

# Chapter 37. 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.

## 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 Chapter 13, “Which protocol can be used?,” on page 39

#### 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. The use of multiple sockets is indicated by a Gateway daemon log message, which is generated when the connection is established.

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

If one or more of the socket connections used for an IPIC connection ends unexpectedly, for example, because of a network error, all the sockets are closed and the IPIC connection is released.

#### Connection sessions

Each IPIC connection can be defined with up to 999 sessions. A single session handles a single request at a time, so that the number of sessions determines the

maximum number of simultaneous requests that can be outstanding over an IPIC connection. The number of sessions is defined on both the CICS server IPCONN definition and the CICS TG IPICSERVER definition. If the number of sessions differs on either end of the connection the actual number of sessions established is negotiated when the connection is established.

**Related information**:

Chapter 14, "Which API can be used?," on page 41
This table shows which APIs are supported over the IPIC, TCP/IP and SNA CICS server connection protocols in local and remote mode.

Chapter 13, "Which protocol can be used?," on page 39
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 393
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 or fully qualified host name of your CICS 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.

See the IPIC connection scenarios for example configurations, or perform the following steps:

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 IBM 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 109.

   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 393.

   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 possible over the connection is determined as follows:

*Table 4. How the number of simultaneous transactions possible over an IPIC connection is determined*

| 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. |

**Note:** For local mode IPIC connections the CICS Transaction Gateway requests 100 send sessions by default. For JEE applications, the number of sessions can be configured using the **ipicSendSessions** connection factory property. For Java base class applications, the number of sessions can be configured using the **CTG_IPIC_SENDSESSIONS** Java property.

Each active session uses one CICS task and 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 244.

## Configuring IPIC on IBM TXSeries

Perform these steps to configure IPIC on IBM TXSeries

### About this task

To enable communication between CICS Transaction Gateway and a TXSeries region, you need to configure a Listener Definition for IPIC over TCP/IP, and optionally, security.

### Procedure

1. Create a Listener Definition (LD) and set the **Protocol** attribute to `IPIC`.
2. Set the **TCPAddress** attribute to define the network adapter addresses. The attribute can be specified as follows:
   - The Internet Protocol (IP) address in dotted decimal notation. For example, `1.23.45.67`. Do not use leading zeros when specifying an address in dotted decimal notation; CICS interprets such an entry as octal.
   - The Internet Protocol (IP) address in dotted hexadecimal notation. For example, `0x01.0x17.0x2D.0x43`.
   - The host name defined in the Internet name service. For example, `aix5.cicsland.com`.

3. Set the **TCPService** attribute to the TCP/IP service name. The TCP/IP service name refers to an entry in the `services` file provided by the operating system. In open systems such as AIX and HP-UX, the services file is in the `/etc/` directory. In Microsoft Windows, the file is in the `C:\windows\system32\drivers\etc` directory. You must add a unique service name and assign a unique TCP port for the service name in the services file. For example:

   ```
   cicsipc1        8598/tcp  #TCP listener1 for region regionName
   ```

   For more information on services files, see your operating system documentation.
4. Set **ActivateOnStartup=**`yes` so the Listener will start when the region starts.

### Example

The following sample command adds an LD entry for IPIC to the CICS region's permanent database:

```
cicsadd -r cics -P -c ld CICSIPC TCPAddress="aix2.cicsland.com"
TCPService="cicsipic" Protocol=IPIC  ActivateOnStartup=yes
```

You can predefine or autoinstall IPIC connections in TXSeries.

### What to do next

To configure security, modify **RemoteSysSecurity=**`verify` in the default CD Entry. For more information, see Providing CICS user security in the TXSeries documentation.

You can configure CICS Systems to communicate over secure channels. For more information, see this link Configuring CICS systems to communicate over secure channels.

To enable link security, follow the instructions in Defining CICS link security

## 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.

The default number of IPIC send sessions in local mode is 100. If you use the ECI resource adapter, you can change this value by setting the **ipicSendSessions** parameter. If you use a Java application, you can change this value by setting the **LOCAL_PROP_IPIC_SENDSESSIONS** property on the JavaGateway object.

By default, CICS TG will send a heartbeat request to CICS every 30 seconds when the IPIC connection is not active if the CICS server supports IPIC heartbeats. If you use the ECI resource adapter, you can change this value or disable heartbeats by setting the **ipicHeartbeatInterval** parameter. If you use a Java application, you can change this value or disable heartbeats by setting the **LOCAL_PROP_IPIC_HEARTBEAT** property on the JavaGateway object. For more information, see CICS Transaction Gateway Base API Programming Reference.

For more information on configuring IPIC in local mode using the ECI resource adapter, refer to "ECI resource adapter deployment parameters" on page 451.

## 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 according to your installation.

To configure an IPIC server definition edit the configuration file directly, see "IPICSERVER section of the configuration file" on page 244 for more information.

**Server name:**

The **SECTION IPICSERVER** parameter defines a logical CICS server name used to reference the actual CICS server. The logical CICS server name is used in API requests.

```
SECTION IPICSERVER=<name>
```

**Description**

Set the value to the name of server that can be used for all requests to access the server from Client applications. The server name can be 1 through 8 characters long. Use characters in the range A through Z, and 0 through 9, and the characters '@', '#', '$', '-'. Lowercase characters, in the range a through z, are converted to uppercase.

This parameter is in the "IPICSERVER section of the configuration file" on page 244.

**Default value**

There is no default value for this parameter.

**Configuration Tool**

In the Configuration Tool, you can set the value of **SECTION IPICSERVER** in the **Server name** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**Description:**

The **description** parameter is optional and can be used to describe the server definition.

**description=<string>**

**Description**

Set the value to a text string. The string can be 1 - 60 characters. Use characters in the range a through z, A through Z, and 0 through 9, and the characters '@', '#', '$', '-'. The description is returned on list systems calls.

This parameter is in the "IPICSERVER section of the configuration file" on page 244 and the "SERVER section of the configuration file" on page 245.

**Default value**

There is no default value for this parameter.

**Configuration Tool**

In the Configuration Tool, you can set the value of **description** in the **Description** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**Network protocol:**

The **protocol** parameter specifies that either the TCP/IP or SNA protocols is used for the server connection.

In the DRIVER section of the configuration file, specify either SECTION DRIVER = SNA or SECTION DRIVER = TCP/IP and specify the correct **drivername** parameter for the connection, for more information, see "DRIVER section of the configuration file" on page 246.

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** parameter is the host name or host IP address of the host on which the CICS server is running.

**hostname=<name|address>**

**Description**
Set the value to the host name or host IP address. Host names are mapped to an IP addresses by either the DNS server or in the `hosts` system file.

This parameter is in the "IPICSERVER section of the configuration file" on page 244.

**Default value**
There is no default value for this parameter.

**Configuration Tool**
In the Configuration Tool, you can set the value of **hostname** in the **Hostname or IP address** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**Port:**

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

**port=<number>**

**Description**
Set the value in the range 1 - 65,535.

This parameter is in the "IPICSERVER section of the configuration file" on page 244.

**Default value**
There is no default value for this parameter.

**Configuration Tool**
In the Configuration Tool, you can set the value of **port** in the **Port** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**IPIC send sessions:**

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

**sendsessions=<number>**

**Description**
Set the value in the range 1 - 999 to specify the number of send sessions. You must also specify the same number in the value in the **receivecount** parameter of the IPCONN definition on the CICS server. If you do not specify a value or specify a lower value in the **receivecount** than the value specified in the **sendsessions** parameter, the value used might be reduced.

This parameter is in the "IPICSERVER section of the configuration file" on page 244.

**Default value**
If this parameter is not specified, the default value is 100.

**Configuration Tool**
In the Configuration Tool, you can set the value of **sendsessions** in the **Server name** field in the **Send sessions** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

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

**Target CICS APPLID:**

The **cicsapplid** parameter identifies the APPLID of the target CICS server.

The **cicsapplid** and **cicsapplidqualifier** parameters can be used to verify that the connection is made to the correct CICS server.

**cicsapplid=<name>**

**Description**
Set a value of up to eight characters, specifying the APPLID of the target CICS server. The target CICS server is identified by the **hostname** and **port** parameters. Setting the **CICSAPPLID** parameter is optional. However, if specified, the **cicsapplid** parameter must match the APPLID of the target CICS server and the **cicsapplidqualifier** parameter must match the network ID of the target CICS server, otherwise the connection is not established.

This parameter is in the "IPICSERVER section of the configuration file" on page 244.

**Default value**
There is no default value for this parameter.

**Configuration Tool**
In the Configuration Tool, you can set the value of **cicsapplid** in the **Target APPLID** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**Target CICS APPLID qualifier:**

The **cicsapplidqualifier** parameter identifies the network ID of the target CICS server.

The **cicsapplid** and **cicsapplidqualifier** parameters can be used to verify that the connection is made to the correct CICS server.

**cicsapplidqualifier=<name>**

**Description**

Set a value of up to eight characters, specifying the network ID of the target CICS server. The target CICS server is identified by the **hostname** and **port** parameters. Setting the **cicsapplidqualifier** is optional. However, if specified, the **cicsapplid** parameter must also be set, the value specified in the **cicsapplid** parameter must match the APPLID of the target CICS server, and the **cicsapplidqualifier** parameter must match the network ID of the target CICS server, otherwise the connection is not established.

This parameter is in the "IPICSERVER section of the configuration file" on page 244.

**Default value**

There is no default value for this parameter.

**Configuration Tool**

In the Configuration Tool, you can set the value of **cicsapplidqualifier** in the **Target APPLID qualifier** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**Connection timeout:**

The **connecttimeout** parameter specifies the maximum period, in seconds, that establishing a CICS server connection is permitted to take.

**connecttimeout=<number>**

**Description**

Set the value in the range 0 - 3600 to specify the time period in seconds. If the value of the **connecttimeout** parameter is set to 0, no limit is set by the Client daemon. A timeout occurs if establishing the connection takes longer than the specified time.

This parameter is in the "IPICSERVER section of the configuration file" on page 244.

**Default value**

If this parameter is not specified, the default value is 0.

**Configuration Tool**

In the Configuration Tool, you can set the value of **connecttimeout** in the **Connection timeout (sec)** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**Server retry interval:**

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

**srvretryinterval=<number>**

**Description**

Set the value in the range 0 - 3600 to specify the time interval in seconds. If

the CICS server that is currently connected becomes disconnected, an attempt is made to reconnect one second after the CICS server becomes disconnected. If the connection attempt fails, additional attempts are made to connect at the interval specified by the **srvretryinterval** parameter. If you specify a value of 0, connection attempts are only initiated if an ECI request is directed at the CICS server and no connection attempt is already in progress. If you specify a value greater than 0, connection attempts are only initiated at the interval specified by the **srvretryinterval** parameter.

This parameter is in the "IPICSERVER section of the configuration file" on page 244.

**Default value**
If this parameter is not specified, the default value is 60 seconds.

**Configuration Tool**
In the Configuration Tool, you can set the value of **srvretryinterval** in the **Server retry interval (sec)** field on the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** page or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**Server idle timeout:**

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

**srvidletimeout=<number>**

**Description**
Set the value in the range 1 - 1080 to specify the period in minutes. A connection is considered idle when there are no outstanding transactions on the CICS server. If an idle connection remains open for the time specified by **srvidletimeout** parameter, the connection is closed. The connection is reestablished when a new a request is sent to the CICS server.

This parameter is in the "IPICSERVER section of the configuration file" on page 244.

**Default value**
If this parameter is not specified, the default value is 0.

**Configuration Tool**
In the Configuration Tool, you can set the value of **srvidletimeout** in the **Server idle timeout (min)** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**Send TCP/IP KeepAlive packets:**

The **tcpkeepalive** parameter determines whether TCP/IP periodically sends keepalive messages to the server to check the connection.

**tcpkeepalive=<Y|N>**

**Description**
Set the value to Y to periodically send keepalive messages.

This parameter is in the "IPICSERVER section of the configuration file" on page 244.

**Default value**
If this parameter is not specified, the default value is Y.

**Configuration Tool**
In the Configuration Tool, you can set the value of **tcpkeepalive** to **Y** by selecting the **Send TCP/IP KeepAlive packets** check box in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**IPIC Heartbeat interval:**

The **heartbeatinterval** parameter specifies the time, in seconds, between heartbeat requests to the CICS server.

**heartbeatinterval=<number>**

**Description**
Set the value in the range 0 - 3600 to specify the heartbeat interval in seconds. Heartbeat requests are small requests which are sent to the CICS server when no other data has arrived on the connection for a period of time, if the CICS server supports heartbeat requests. If a response to a heartbeat request is not received within the heartbeat interval, the connection is closed, and if a server retry interval is configured, then an attempt will be made to reconnect after 1 second. To disable heartbeat requests, set the value of the **heartbeatinterval** parameter to 0.

This parameter is in the "IPICSERVER section of the configuration file" on page 244.

**Default value**
If this parameter is not specified, the default value is 30 seconds.

**Configuration Tool**
In the Configuration Tool, you can set the value of **heartbeatinterval** in the **Heartbeat Interval (sec)** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** page or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**ECI timeout:**

The **ecitimeout** parameter specifies the maximum period, in seconds, that CICS Transaction Gateway waits before an ECI request times out.

**ecitimeout=<number>**

**Description**
Set the value in the range 0 - 32767 to specify the time period in seconds for a response to be received for an ECI request. A timeout occurs when no response is received from the CICS server in the specified time. If a timeout occurs, the client does not receive confirmation from CICS that a

unit of work is backed out or committed. The **ecitimeout** parameter overrides the ECI timeout value set on an ECI request. If the value of the **ecitimeout** parameter is set to 0, the ECI timeout values specified by the ECI requests are used.

This parameter is in the "IPICSERVER section of the configuration file" on page 244.

**Default value**
If this parameter is not specified, the default value is 0.

**Configuration Tool**
In the Configuration Tool, you can set the value of **ecitimeout** in the **ECI timeout (sec)** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**Use SSL:**

The **ssl** parameter controls the use of SSL over an IPIC connection.

**ssl=<Yes|No>**

**Description**
Set the value to Yes to use SSL for this IPIC connection.

**Default value**
The default value is No.

**Use only these ciphers:**

Use the **ciphersuites** parameter to restrict the set of cipher suites that can be used with the SSL protocol.

**ciphersuites=<name>**

**Description**
Specify the cipher suites that CICS Transaction Gateway client applications can use to connect to CICS Transaction Server for z/OS. You can define multiple cipher suites by separating them with a comma. If none of the cipher suites listed are supported by the JSSE provider, CICS Transaction Gateway fails to start. If no cipher suite is specified or the parameter is omitted, all available cipher suites can be used.

This parameter is in the IPICSERVER section of the configuration file.

**Default value**
If this parameter is not specified, the default is that all available cipher suites are available.

**Configuration Tool**
In the Configuration Tool, you can set the value of **ciphersuites** in the **Use only these ciphers** field in the **Server connection page** page. This page can be accessed by selecting an existing server in the CICS Servers section of the CICS Transaction Gateway pane or by creating a server definition. To create a server definition, click **Options** > **New Server**or click the **New Server** icon.

Enter the cipher suite name in the field, and then click **Add** to add it to the list. To remove a cipher suite, select the suite in the list and click **Remove**.

# 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 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 in the *CICS Transaction Gateway for Multiplatforms: Developing Applications*.
3. Set the values according to your installation.

To configure a TCP/IP server definition by editing the configuration file directly, see "SERVER section of the configuration file" on page 245.

## Server name

The **SECTION SERVER** parameter defines a logical CICS server name used to reference the actual CICS server. The logical CICS server name is used in API requests.

**SECTION SERVER=<name>**

**Description**

Set the value to a logical CICS server name. The server name can be 1 - 8 characters long. Use characters in the range A through Z, and 0 through 9, and the characters '@', '#', '$', '-'. Lowercase characters, in the range a through z, are converted to uppercase.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**

There is no default value for this parameter.

**Configuration Tool**

In the Configuration Tool, you can set the value of **SECTION SERVER** in the **Server name** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

## Description

The **description** parameter is optional and can be used to describe the server definition.

**description=<string>**

**Description**

Set the value to a text string. The string can be 1 - 60 characters. Use characters in the range a through z, A through Z, and 0 through 9, and the characters '@', '#', '$', '-'. The description is returned on list systems calls.

This parameter is in the "IPICSERVER section of the configuration file" on page 244 and the "SERVER section of the configuration file" on page 245.

**Default value**

There is no default value for this parameter.

**Configuration Tool**

In the Configuration Tool, you can set the value of **description** in the **Description** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

## Network protocol

The **protocol** parameter specifies that either the TCP/IP or SNA protocols is used for the server connection.

In the DRIVER section of the configuration file, specify either SECTION DRIVER = SNA or SECTION DRIVER = TCP/IP and specify the correct **drivername** parameter for the connection, for more information, see "DRIVER section of the configuration file" on page 246.

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 the initial transaction identifier of the transaction that is run when the terminal emulator connects to the server.

**INITIALTRANSID=<name>**

**Description**

Set the value to the name of initial transaction identifier. The name must be 1 - 128 characters long and is case sensitive. The first four characters, or the characters before the first blank space in the string, are taken as the transaction identifier. The remaining characters are passed to the transaction when the transaction is invoked. The transaction must not require terminal input.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**

There is no default value for this parameter.

**Configuration Tool**

In the Configuration Tool, you can set the value of **INITIALTRANSID** in the **Initial transation** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

## Model terminal definition

The **MODELTERM** parameter specifies the name of a model terminal definition for the CICS server.

**MODELTERM=<name>**

**Description**

Set the value to the name of model terminal definition that identifies the characteristics of terminals to be automatically installed. The name must be 1 - 16 characters long and is case sensitive. The interpretation of the **MODELTERM** parameter is server-specific.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**

If this parameter is not specified, a default definition is used. This default is server-specific.

**Configuration Tool**

In the Configuration Tool, you can set the value of **MODELTERM** in the **Model terminal definition** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

## CICS server connection protocols

The **protocol** parameter defines the CICS server connection protocol used to connect to your CICS servers

**protocol=<protocol>**

**Description**

Set the value to either SNA or TCPIP.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**

There is no default value for this parameter.

**Configuration Tool**

This parameter cannot be set using the Configuration Tool.

## Host name or IP address

The **NETNAME** parameter is the host name or host IP address of the host machine on which the CICS server is running.

**NETNAME=<name|number>**

**Description**

Set the value to the host name or host IP address. Host names are mapped to IP addresses by either the DNS server or in the `hosts` system file. For Windows, the `hosts` system file is in the `%windir%\system32\drivers\etc` directory. For UNIX and Linux, the `hosts` system file is in the `/etc` directory. Specifying a host name in the **NETNAME** parameter can prevent issues if the host IP address changes.

This parameter is in the "SERVER section of the configuration file" on page 245.

The host machine might cache the contents of the hosts system file and require manual intervention or a system restart to refresh the cache. See the documentation for the operating system for further information.

**Default value**

There is no default value for this parameter.

**Configuration Tool**

In the Configuration Tool, you can set the value of **NETNAME** in the **Hostname or IP address** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

## Port

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

**PORT=<number>**

**Description**

Set the value in the range 0 - 65,535. If you specify a value of 0, the `services` system file is used to locate the port number for the CICS service using the TCP protocol. If an entry cannot be found in the `services` system file, a value of 1435 is used. Port 1435 is the number of the port assigned

to the Client daemon in the TCP/IP architecture. On Windows platforms, the `services` system file is in the `%windir%\system32\drivers\etc` directory and on UNIX and Linux platforms, it is in the `/etc` directory.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**

If this parameter is not specified, the default value is 0.

**Configuration Tool**

In the Configuration Tool, you can set the value of **PORT** in the **Port** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

## Connection timeout

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

**CONNECTTIMEOUT=<number>**

**Description**

Set the value in the range 0 - 3600 to specify the time period in seconds. If the value of the **CONNECTTIMEOUT** parameter is set to 0, no limit is set by the Client daemon. A timeout occurs if establishing the connection takes longer than the specified time. If a timeout occurs, the TCP/IP socket is closed and either return code ECI_ERR_NO_CICS or CICS_EPI_ERR_FAILED is passed back to the Client application.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**

If this parameter is not specified, the default value is 0.

**Configuration Tool**

In the Configuration Tool, you can set the value of **CONNECTTIMEOUT** in the **Connection timeout (sec)** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

## Server idle timeout

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

**SRVIDLETIMEOUT=<number>**

**Description**

Set the value in the range 1 - 1080 to specify the period in minutes. The period starts when the number of outstanding units of work on the connection is zero. If a server connection times out, the Client daemon behaves in the same way as when the `cicscli -x=<servername>` command is issued and the value of the **srvretryinterval** parameter is ignored. The connection is automatically reestablished when a Client application sends the next ECI, EPI, or ESI request and the **srvretryinterval** parameter is re-enabled.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**
If this parameter is not specified, the default value is 0.

**Configuration Tool**
In the Configuration Tool, you can set the value of **SRVIDLETIMEOUT** in the **Server idle timeout (min)** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

## Send TCP/IP KeepAlive packets

The **TCPKEEPALIVE** parameter determines whether TCP/IP periodically sends keepalive messages to the server to check the connection.

**TCPKEEPALIVE=<Y|N>**

**Description**
Set the value to Y to periodically send keepalive messages.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**
If this parameter is not specified, the default value is N.

**Configuration Tool**
In the Configuration Tool, you can set the value of **TCPKEEPALIVE** to **Y** by selecting the **Send TCP/IP KeepAlive packets** check box in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

## Use uppercase security

The **UPPERCASESECURITY** parameter specifies that user IDs and passwords are converted to uppercase before passing to the CICS server.

**UPPERCASESECURITY=<Y|N>**

**Description**
Set the value to Y to enable uppercase conversions. If the **UPPERCASESECURITY** is set to N, user IDs and passwords retain any lowercase characters.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**
The default for this parameter is Y.

**Configuration Tool**
In the Configuration Tool, you can set the value of **UPPERCASESECURITY** to **Y** by selecting the **Use upper case security** check box in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

# Configuring SNA

Perform these steps to configure an SNA server connection.

## Configuring SNA on UNIX and Linux

Perform these steps to configure a SNA server connection on UNIX and Linux.

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).

You need to install and configure an SNA communications server, such as IBM Communications Server. To set up communication over SNA define the following settings 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 connections support data synchronization levels (sync levels) 0 and 1.

### Overview of SNA on UNIX and Linux: configuration definitions

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

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

*Table 5. Matching definitions for SNA*

| IBM 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 |

*Table 5. Matching definitions for SNA (continued)*

| IBM VTAM | CICS Transaction Server | SNA Server | ctg.ini | Example |
|---|---|---|---|---|
| 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 IBM VTAM network name. It is defined for your IBM VTAM network in the IBM VTAM start options.
2. The PU is the name of the Communications Server physical unit (PU). It is named in the IBM VTAM switched major node.
3. The LU is the independent LU6.2 used by the CICS Transaction Gateway. It must also be defined to IBM VTAM in a switched Major Node.
4. The XID (or node ID) is configured in the IBM 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 IBM VTAM APPL. It is defined in the IBM 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 IBM VTAM logon mode table, which must be named on the IBM VTAM APPL definition.
8. The Host system control point name is the CP for the PU of the IBM 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.

You need to set these environment variables:

**LD_LIBRARY_PATH**
A list of directory names, separated by colons, containing the dynamically linked libraries that are to be used when the program is run.

**LD_RUN_PATH**
A list of directory names, separated by colons, containing the dynamically linked libraries that are to be used when the program is link edited,

**LD_PRELOAD**
The name of a shared library object that is to be loaded before any other shared libraries are searched.

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 on UNIX and Linux:

1. Start the administration application **xsnapadmin** on HP-UX or **xsnaadmin** on all other UNIX and Linux platforms
2. Click **Service** > **APPC** > **Transaction Programs**.
3. Select **TP invocation**.
4. Click **New**.
5. Enter `CRSR` as the `Application TP`.
6. Set **Parameters are for invocation on any LU** and **Queue incoming allocates**
7. In the field **Full path to TP executable**, enter the path to the executable file as: `<install_path>/bin/cclclnt`.
8. Set **Arguments** to `CRSR`.
9. Set **Userid** and **Group** to the user ID and group that the application will run under.
10. Click **OK** to close the TP definition window and save your parameters.
11. Click **Local LU** > **Properties** > **Advanced** and ensure that the **Local LU advanced** parameter is set to the Attach routing computer host name. The Attach routing computer name option is only available when the Communications Server is running as part of a client/server domain.

### Configuring for ATI with IBM Remote API Client (split SNA stack)

To use ATIs, the Communications Server must know the TCP/IP address of the connecting Remote API Client machine and the Remote API Client must have a definition for the transaction CRSR. The Remote API Client machine should have a static IP address. If it has a dynamic IP address, the connection might need to be re-configured if the dynamic IP address changes.

1. On the machine where Communication Server is installed:
   a. Start the administration application **xsnaadmin** on UNIX or Linux platforms.
   b. Click **Independent local LU** > **Advanced**.
   c. In `Local LU parameters` section, in the `Computer` field, specify the TCP/IP address of the computer that CICS TG is installed on.
2. On the IBM Remote API Client on the machine where CICS TG is installed:
   a. Start the IBM Remote API Client Configuration Utility
   b. In the Customize Options panel specify the:
      - `IBM Remote API Domain Name` : The domain name of the IBM Remote API LAN (displayed at the top of the Communications Server administration application)
      - `IBM Remote API Server Name` : The name of the Communications server node (displayed in the Communications Server administration application) defined to communicate with CICS in step 1.
3. In the directory where the IBM Remote API Client is installed create an ASCII file containing the following text:

   ```
   [CRSR]
   TIMEOUT=-1
   TYPE=QUEUED
   PATH=<CICS_TG_install_dir>/bin/cclclnt
   SHOW=NORMAL
   SECURITY_TYPE=APPLICATION
   USERID=<user_ID>
   ```

   where `<user_ID>` is the user ID that is used to start the TP.
4. Issue the command : **tpinst32 -a filename**.

For full details of the configuration options see the *IBM Communications Server for Data Center Deployment Administration Guide*.

## 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" on page 135.

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 IBM 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 134), 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 135, 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 41 for CICS TS 4.1 (`CICSTS41.CICS.SDFHSAMP`), 42 for CICS TS 4.2, and 51 for CICS TS 5.1.

## 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: Developing Applications*.
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 245.

**Server name:**

The **SECTION SERVER** parameter defines a logical CICS server name used to reference the actual CICS server. The logical CICS server name is used in API requests.

**SECTION SERVER=<name>**

**Description**
Set the value to a logical CICS server name. The server name can be 1 - 8 characters long. Use characters in the range A through Z, and 0 through 9, and the characters '@', '#', '$', '-'. Lowercase characters, in the range a through z, are converted to uppercase.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**
There is no default value for this parameter.

**Configuration Tool**
In the Configuration Tool, you can set the value of **SECTION SERVER** in the **Server name** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**Description:**

The **description** parameter is optional and can be used to describe the server definition.

**description=<string>**

**Description**
Set the value to a text string. The string can be 1 - 60 characters. Use characters in the range a through z, A through Z, and 0 through 9, and the characters '@', '#', '$', '-'. The description is returned on list systems calls.

This parameter is in the "IPICSERVER section of the configuration file" on page 244 and the "SERVER section of the configuration file" on page 245.

**Default value**
There is no default value for this parameter.

**Configuration Tool**
In the Configuration Tool, you can set the value of **description** in the **Description** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**Network protocol:**

The **protocol** parameter specifies that either the TCP/IP or SNA protocols is used for the server connection.

In the DRIVER section of the configuration file, specify either SECTION DRIVER = SNA or SECTION DRIVER = TCP/IP and specify the correct **drivername** parameter for the connection, for more information, see "DRIVER section of the configuration file" on page 246.

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 the initial transaction identifier of the transaction that is run when the terminal emulator connects to the server.

**INITIALTRANSID=<name>**

**Description**
Set the value to the name of initial transaction identifier. The name must be 1 - 128 characters long and is case-sensitive. The first 4 characters, or the characters before the first blank space in the string, are taken as the transaction identifier. The remaining characters are passed to the transaction when the transaction is invoked. The transaction must not require terminal input.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**
There is no default value for this parameter.

**Configuration tool**
In the Configuration tool, you can set the value of **INITIALTRANSID** in the **Initial transaction** field in the **Server connection** page. The page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**Model terminal definition:**

The **MODELTERM** parameter specifies the name of a model terminal definition for the CICS server.

**MODELTERM=<name>**

**Description**
Set the value to the name of model terminal definition that identifies the characteristics of terminals to be automatically installed. The name must be 1 - 16 characters long and is case sensitive. The interpretation of the **MODELTERM** parameter is server-specific.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**
If this parameter is not specified, a default definition is used. This default is server-specific.

**Configuration Tool**
In the Configuration Tool, you can set the value of **MODELTERM** in the **Model terminal definition** field in the **Server connection** page. This page can be

accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**CICS server connection protocols:**

The **protocol** parameter defines the CICS server connection protocol used to connect to your CICS servers.

**protocol=<protocol>**

**Description**
Set the value to either SNA or TCPIP.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**
There is no default value for this parameter.

**Configuration Tool**
This parameter cannot be set using the Configuration Tool.

**Local LU name:**

The **LOCALLUNAME** parameter specifies the local LU name alias as defined on the SNA Communications Server.

**LOCALLUNAME=<name>**

**Description**
Set the value to the local LU name alias.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**
There is no default value for this parameter.

**Configuration Tool**
Set the value of **LOCALLUNAME** in the **Local LU name alias** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**Partner LU name:**

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

**NETNAME=<name>**

**Description**
Set the value to the LU name of the CICS server. The value must be a qualified name of up to 17 characters. You can have an alias **Partner LU name** only when the **Local LU name** is an alias.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**
There is no default value for this parameter.

**Configuration Tool**
In the Configuration Tool, you can set the value of **NETNAME** in the **Partner LU name** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**Use Partner LU alias name:**

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

**PARTNERLUALIAS=<Y>**

**Description**
Set the value to Y to specify that the **Partner LU name** is an alias. Do not specify this parameter if the **Partner LU name** is not an alias.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**
The default for this parameter is Y.

**Configuration Tool**
In the Configuration Tool, you can set the value of **PARTNERLUALIAS** to **Y** by selecting the **Use Partner LU alias name** check box in the **Server name** field in the **Server connection** page. The **Use Partner LU alias name** check box is available only if **Use Local LU alias name** check box is checked. The **Server connection** page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**Mode name:**

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

**MODENAME=<name>**

**Description**
Set the value to the name of the mode. Enter * if you want the mode name to be filled with spaces.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**
There is no default value for this parameter.

**Configuration Tool**
In the Configuration Tool, you can set the value of **MODENAME** in the **Mode name** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**Server idle timeout:**

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

**SRVIDLETIMEOUT=<number>**

**Description**
Set the value in the range 1 - 1080 to specify the period in minutes. The period starts when the number of outstanding units of work on the connection is zero. If a server connection times out, the Client daemon behaves in the same way as when the `cicscli -x=<servername>` command is issued and the value of the **srvretryinterval** parameter is ignored. The connection is automatically reestablished when a Client application sends the next ECI, EPI, or ESI request and the **srvretryinterval** parameter is re-enabled.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**
If this parameter is not specified, the default value is 0.

**Configuration Tool**
In the Configuration Tool, you can set the value of **SRVIDLETIMEOUT** in the **Server idle timeout (min)** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**Use uppercase security:**

The **UPPERCASESECURITY** parameter specifies that user IDs and passwords are converted to uppercase before passing to the CICS server.

**UPPERCASESECURITY=<Y|N>**

**Description**
Set the value to Y to enable uppercase conversions. If the **UPPERCASESECURITY** is set to N, user IDs and passwords retain any lowercase characters.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**
The default for this parameter is Y.

**Configuration Tool**
In the Configuration Tool, you can set the value of **UPPERCASESECURITY** to **Y** by selecting the **Use upper case security** check box in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

## Configuring SNA on Windows

Perform these steps to configure a SNA server connection on Windows.

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).

You need to install and configure an SNA communications server, such as IBM Communications Server. To set up communication over SNA define the following settings 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 connections support data synchronization levels (sync levels) 0 and 1.

More information on connecting CICS Transaction Gateway to CICS Transaction Server for z/OS, using Host Integration Server, is in *CICS Transaction Gateway V5 The WebSphere Connector for CICS Redbook*.

### Overview of SNA on Windows: configuration definitions

An overview of how to configure SNA including a table of matching definitions.

Configuration requires action on IBM VTAM, CICS, the CICS Transaction Gateway, and one of the following:

- IBM Personal Communications
- IBM Communications Server
- Microsoft Host Integration Server

Install and configure only one of these products using the documentation supplied with the product. If you have more than one product installed on your machine, you might experience unexpected results. The following table shows the entries that you need to make. The values in the Example column are taken from the sample configuration documents.

*Table 6. Matching definitions for SNA*

| IBM VTAM | CICS | IBM Personal Communications | IBM Communications Server | Microsoft Host Integration Server | Configuration file | Example |
|---|---|---|---|---|---|---|
| NETID | — | First part of Partner LU name | First part of Partner LU name | Network Name | Partner LU name | ABC3XYZ4 |
| PU | — | CP name | CP name | Control Point Name | — | IYAMR004 |
| LU | Netname | Local LU alias on Windows or name on UNIX and Linux | Local LU alias on Windows or name on UNIX and Linux | LU Name | Local LU name | IYAMT040 |
| XID | — | Local Node ID | Local Node ID | Local Node ID | — | 05D 316DF |
| Token Ring destination address | — | Destination address | Destination address | Remote Network Address | — | 400045121088 |
| APPL | Applid | Second part of Partner LU name | Second part of Partner LU name | Remote LU Name | Partner LU name | IYCKCTU1 |
| LogMode | Modename | Mode name | Mode name | Mode Name | Mode name | LU62PS |
| SSCP | — | Adjacent CP name | — | — | — | — |

Ensure that the SNA server and SNA client, if you are using one, are started before starting the CICS Transaction Gateway.

## Configuring IBM Communications Server

Run SNA Node Configuration to configure SNA support, define a connection and the local LU, partner LU, and mode specified in the configuration file. Configure the local LU to be independent.

If you are using a Communications Server SNA Client, you must select the **SNA API client use** check box in the Local LU 6.2 definition panel.

To enable ATI against CICS Client terminals, define the transaction program CRSR on the Communications Server, whether or not a Communications Server SNA Client is being used:

**TP name**
CRSR

**Service TP**
No

**Complete path name**
`<install_path>\bin\cclclnt.exe`

**Program parameters**
CRSR

**Conversation type**
Mapped

**Synchronization level**
Any

**Conversation security required**
No

**Receive_Allocate timeout**
0

**Incoming allocate timeout**
Default

**TP instance limit**
1

**PIP allowed**
No

**SNA Client**
Yes/No

**Dynamically loaded**
No

**Full duplex support**
No

**Configuring IBM Communications Server Client:**

To set up the Communications Server Windows (client) you must complete these actions.

1. Run **Configuration** to configure Global Data and the LU6.2 Server List.
2. To enable ATI against CICS Client terminals, define the transaction program CRSR on the SNA Client:

   **TP name**
   CRSR

   **LU alias**
   Alias of the local LU defined for the connection

   **Complete path name**
   `<install_path>\bin\cclclnt.exe`

   Your `<install_path>` must not contain spaces; use short file and directory names.

   **Program operation**
   Attach Manager Started

   **Program parameters**
   CRSR

   **Service TP**
   No

   **Attach Manager Started**
   No
3. Start the Attach Manager.

## Configuring for ATI

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

To define the transaction program:

1. Start the administration application **xsnaadmin**.
2. Click **Service** > **APPC** > **Transaction Programs**.
3. Select **TP invocation**.
4. Click **Add**.
5. Enter `CRSR` as the `Application TP`.
6. Set **Parameters are for invocation on any LU** and **Queue incoming allocates**

7. In the field **Full path to TP executable**, enter the path to the executable file as: `<install_path>/bin/cclclnt`.
8. Set **Arguments** to `CRSR`.
9. Set **Userid** and **Group** to the user ID and group that the application will run under.
10. Click **OK** to close the TP definition window and save your parameters.
11. Click **Local LU** > **Properties** > **Advanced** and ensure that the **`Local LU advanced`** parameter is set to the Attach routing computer host name. The Attach routing computer name option is only available when the Communications Server is running as part of a client/server domain.

### Configuring for ATI with IBM Remote API Client (split SNA stack)

To use ATIs, the Communications Server must know the TCP/IP address of the connecting Remote API Client machine and the Remote API Client must have a definition for the transaction CRSR. The Remote API Client machine should have a static IP address. If it has a dynamic IP address, the connection might need to be re-configured if the dynamic IP address changes.

1. On the machine where Communication Server is installed:
   a. Start the administration application **`xsnaadmin`**.
   b. Click **Independent local LU** > **Advanced**.
   c. In `Local LU parameters` section, in the `Computer` field, specify the TCP/IP address of the computer that CICS TG is installed on.
2. On the IBM Remote API Client on the machine where CICS TG is installed:
   a. Start the IBM Remote API Client Configuration Utility
   b. In the Customize Options panel specify the:
      - `IBM Remote API Domain Name` : The domain name of the IBM Remote API LAN (displayed at the top of the Communications Server administration application)
      - `IBM Remote API Server Name` : The name of the Communications server node (displayed in the Communications Server administration application) defined to communicate with CICS in step 1.
3. In the directory where the IBM Remote API Client is installed create an ASCII file containing the following text:

   ```
   [CRSR]
   TIMEOUT=-1
   TYPE=QUEUED
   PATH=<install_path>/bin/cclclnt.exe
   SHOW=NORMAL
   SECURITY_TYPE=APPLICATION
   USERID=<domain_name>/<user_ID>
   ```

   where `<domain_name>/<user_ID>` is the domain and user ID that the client uses to start the TP.
4. Issue the command : **`tpinst32 -a filename`**.

For full details of the configuration options see the *IBM Communications Server for Data Center Deployment Administration Guide*.

## Configuring IBM Personal Communications

This topic provides an overview of installing, setting up, and configuring IBM Personal Communications.

To configure and run IBM Personal Communications carry out the following steps:

1. Run SNA Node Configuration to configure SNA support.
2. Define a node, a device, and a connection. Use an adjacent CP type of APPN Node or backlevel LEN in the connection definition.
3. Define the local LU, partner LU, and mode specified in the configuration file. Configure the local LU to be independent.
4. Start the node from SNA Node Operations before starting a connection from the Client daemon to a server that uses Personal Communications.
5. To enable ATI against CICS client terminals, define the transaction program CRSR:

**TP name**
CRSR

**Service TP**
No

**Complete path name**
`<install_path>\bin\cclclnt.exe`

Your `<install_path>` must not contain spaces; use short file and directory names.

**Program parameters**
CRSR

**Conversation type**
Mapped

**Synchronization level**
Any

**Conversation security required**
No

**Receive_Allocate timeout**
0

**Incoming allocate timeout**
Default

**TP instance limit**
1

**PIP allowed**
No

**Dynamically loaded**
No

**Full duplex support**
No

### Running Personal Communications

Start the IBM Personal Communications node before you start an SNA connection from the Client daemon.

## Configuring Microsoft Host Integration Server (SNA Server)

This topic provides an overview of installing, setting up, and configuring Microsoft Host Integration Server (SNA Server).

### Installing Microsoft Host Integration Server (SNA Server)

1. Install Host Integration Server client on a machine that contains the CICS Transaction Gateway, which in turn communicates with the CICS server through the Host Integration Server machine. For information on installing and configuring Microsoft Host Integration Server and Host Integration Server clients, see the Microsoft Host Integration Server documentation.
2. To allow workstations using the Windows SNA Client to communicate with other systems, define the LUs used by the client systems as Local APPC LUs to the Host Integration Server workstation.

### Setting up the Host Integration Server Windows client

1. On the Host Integration Server Windows client, enter the following:

   **SNA client mode**
   Remote

   **Primary Server**
   Name of the Host Integration Server workstation that the client systems using SNA Client will use. This computer name of the Host Integration Server is displayed in both the Host Integration Server Admin **Servers and Connections** window, and in the **Network Settings** control panel dialog box.
2. Run SNABASE.EXE to start the Host Integration Server Client SnaBase. If you do not, the CICS Transaction Gateway cannot communicate with the Host Integration Server.
3. Start the Host Integration Server Windows client.

### Configuration documentation

More information on connecting CICS Transaction Gateway to CICS Transaction Server for z/OS, using Host Integration Server, is in *CICS Transaction Gateway V5 The WebSphere Connector for CICS*.

## 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" on page 135.

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 IBM 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 134), 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 135, 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 41 for CICS TS 4.1 (`CICSTS41.CICS.SDFHSAMP`), 42 for CICS TS 4.2, and 51 for CICS TS 5.1.

## 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: Developing Applications*.
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 245.

**Server name:**

The **SECTION SERVER** parameter defines a logical CICS server name used to reference the actual CICS server. The logical CICS server name is used in API requests.

**SECTION SERVER=<name>**

**Description**

Set the value to a logical CICS server name. The server name can be 1 - 8 characters long. Use characters in the range A through Z, and 0 through 9, and the characters '@', '#', '$', '-'. Lowercase characters, in the range a through z, are converted to uppercase.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**

There is no default value for this parameter.

**Configuration Tool**

In the Configuration Tool, you can set the value of **SECTION SERVER** in the **Server name** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**Description:**

The **description** parameter is optional and can be used to describe the server definition.

**description=<string>**

**Description**

Set the value to a text string. The string can be 1 - 60 characters. Use characters in the range a through z, A through Z, and 0 through 9, and the characters '@', '#', '$', '-'. The description is returned on list systems calls.

This parameter is in the "IPICSERVER section of the configuration file" on page 244 and the "SERVER section of the configuration file" on page 245.

**Default value**

There is no default value for this parameter.

**Configuration Tool**

In the Configuration Tool, you can set the value of **description** in the **Description** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**Network protocol:**

The **protocol** parameter specifies that either the TCP/IP or SNA protocols is used for the server connection.

In the DRIVER section of the configuration file, specify either SECTION DRIVER = SNA or SECTION DRIVER = TCP/IP and specify the correct **drivername** parameter for the connection, for more information, see "DRIVER section of the configuration file" on page 246.

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 the initial transaction identifier of the transaction that is run when the terminal emulator connects to the server.

**INITIALTRANSID=<name>**

**Description**
Set the value to the name of initial transaction identifier. The name must be 1 - 128 characters long and is case-sensitive. The first 4 characters, or the characters before the first blank space in the string, are taken as the transaction identifier. The remaining characters are passed to the transaction when the transaction is invoked. The transaction must not require terminal input.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**
There is no default value for this parameter.

**Configuration tool**
In the Configuration tool, you can set the value of **INITIALTRANSID** in the **Initial transaction** field in the **Server connection** page. The page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**Model terminal definition:**

The **MODELTERM** parameter specifies the name of a model terminal definition for the CICS server.

**MODELTERM=<name>**

**Description**
Set the value to the name of model terminal definition that identifies the characteristics of terminals to be automatically installed. The name must be 1 - 16 characters long and is case sensitive. The interpretation of the **MODELTERM** parameter is server-specific.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**
If this parameter is not specified, a default definition is used. This default is server-specific.

**Configuration Tool**
In the Configuration Tool, you can set the value of **MODELTERM** in the **Model terminal definition** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**CICS server connection protocols:**

The **protocol** parameter defines the CICS server connection protocol used to connect to your CICS servers.

**protocol=<protocol>**

**Description**
Set the value to either SNA or TCPIP.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**
There is no default value for this parameter.

**Configuration Tool**
This parameter cannot be set using the Configuration Tool.

**Local LU name:**

The **LOCALLUNAME** parameter specifies the local LU name as defined on the SNA Communications Server.

**LOCALLUNAME=<name>**

**Description**
Set the value to the local LU name.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**
There is no default value for this parameter.

**Configuration Tool**
In the Configuration Tool, you can set the value of **LOCALLUNAME** in the **Local LU name** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**Use local LU alias name:**

The **LOCALLUALIAS** parameter specifies that the local LU name can be used by the CICS Transaction Gateway on Windows as an alias.

**LOCALLUALIAS=<Y|N>**

**Description**
Set the value to Y to use the local LU name as an alias. An alias name is mandatory if you are configuring the CICS Transaction Gateway on Windows to communicate with IBM Communications Server running on UNIX or Linux. Either a local LU name or a local LU alias name is mandatory when CICS Transaction Gateway on Windows communicates with IBM Communications Server for Windows.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**
If this parameter is not specified, the default value is N.

**Configuration Tool**
In the Configuration Tool, you can set the value of **LOCALLUALIAS** to **Y** by selecting the **Use Local LU alias name** check box in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**Partner LU name:**

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

**NETNAME=<name>**

**Description**
Set the value to the LU name of the CICS server. The value must the alias name and it must be eight characters or less. You can have an alias **Partner LU name** only when the **Local LU name** is an alias.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**
There is no default value for this parameter.

**Configuration Tool**
In the Configuration Tool, you can set the value of **NETNAME** in the **Partner LU name** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**Use Partner LU alias name:**

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

**PARTNERLUALIAS=<Y>**

**Description**
Set the value to Y to specify that the **Partner LU name** is an alias. Do not specify this parameter if the **Partner LU name** is not an alias.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**
The default for this parameter is not specified.

**Configuration Tool**
In the Configuration Tool, you can set the value of **PARTNERLUALIAS** to **Y** by selecting the **Use Partner LU alias name** check box in the **Server name** field in the **Server connection** page. The **Use Partner LU alias name** check box is available only if **Use Local LU alias name** check box is checked. The **Server connection** page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by

creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**Mode name:**

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

**MODENAME=<name>**

**Description**
Set the value to the name of the mode. Enter * if you want the mode name to be filled with spaces.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**
There is no default value for this parameter.

**Configuration Tool**
In the Configuration Tool, you can set the value of **MODENAME** in the **Mode name** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**Server idle timeout:**

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

**SRVIDLETIMEOUT=<number>**

**Description**
Set the value in the range 1 - 1080 to specify the period in minutes. The period starts when the number of outstanding units of work on the connection is zero. If a server connection times out, the Client daemon behaves in the same way as when the `cicscli -x=<servername>` command is issued and the value of the **srvretryinterval** parameter is ignored. The connection is automatically reestablished when a Client application sends the next ECI, EPI, or ESI request and the **srvretryinterval** parameter is re-enabled.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**
If this parameter is not specified, the default value is 0.

**Configuration Tool**
In the Configuration Tool, you can set the value of **SRVIDLETIMEOUT** in the **Server idle timeout (min)** field in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

**Use uppercase security:**

The **UPPERCASESECURITY** parameter specifies that user IDs and passwords are converted to uppercase before passing to the CICS server.

**UPPERCASESECURITY=<Y|N>**

**Description**

Set the value to Y to enable uppercase conversions. If the **UPPERCASESECURITY** is set to N, user IDs and passwords retain any lowercase characters.

This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**

The default for this parameter is Y.

**Configuration Tool**

In the Configuration Tool, you can set the value of **UPPERCASESECURITY** to **Y** by selecting the **Use upper case security** check box in the **Server connection** page. This page can be accessed by selecting an existing server in the **CICS Servers** section of the **CICS Transaction Gateway** pane or by creating a server definition. To create a server definition, click **Options** > **New Server** or click the **New Server** icon.

# Chapter 38. Configuring Gateway daemon settings

The Gateway daemon settings are used for remote mode scenarios and are defined in the `ctg.ini` configuration file. 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 determines the number of connection manager threads that are created on startup that are available for client connections.

**initconnect=<number>**

**Description**

Set the value in the range 1 - 1,000,000 to specify the initial number of connection manager threads. This value should equal the usual number of JavaGateway objects opened by all connected clients. However, you might need to set this number to less than the maximum supported value because of constraints on memory or other system resources.

You can use the **-initconnect** option to override the value of the **initconnect** parameter. For more information, for Windows see, "ctgservice command reference" on page 365, and for UNIX and Linux see, "ctgstart command reference" on page 369.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**

If this parameter is not specified, the default value is 1.

**Configuration Tool**

In the Configuration Tool, you can set the value of **initconnect** in the **Initial number of connection manager threads** field in the **Resources** tab of the **Gateway daemon settings** page.

### Maximum number of connection manager threads

The **maxconnect** parameter defines the maximum number of JavaGateway objects that can be opened at any one time by all the remotely connected Client applications.

**maxconnect=<number>**

**Description**

Set the value in the range 1-1,000,000 to specify the maximum number of connection manager threads. You might need to set this value to less than the supported maximum value because of constraints on memory or other system resources. You can specify that there is no limit to the number of connection manager threads by setting the value of **maxconnect** to -1. For more information about threading limits, see "Threading model" on page 417.

You can use the **-maxconnect** option to override the value of the **maxconnect** parameter. For more information, for Windows see, "ctgservice command reference" on page 365, and for UNIX and Linux see, "ctgstart command reference" on page 369.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**
If this parameter is not specified, the default value is 100.

**Configuration Tool**
In the Configuration Tool, you can set the value of **maxconnect** in the **Maximum number of Connection Manager threads** field in the **Resources** tab of the **Gateway daemon settings** page. You can also set the value of **maxconnect** to -1 by selecting the **Unrestricted** check box associated with the **Maximum number of Connection Manager threads** field.

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

## Initial number of worker threads

The **initworker** parameter determines the number of worker threads that are created on startup that are available for processing client requests.

**initworker=<number>**

**Description**
Set the value in the range 1 - 1,000,000 to specify the initial number of worker threads. This value should equal the number of concurrent requests that the Gateway daemon is expected to process. However, you might need to set this number to less than the maximum supported value because of constraints on memory or other system resources.

You can use the **-initworker** option to override the value of the **initworker** parameter. For more information, for Windows see, "ctgservice command reference" on page 365, and for UNIX and Linux see, "ctgstart command reference" on page 369.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**
If this parameter is not specified, the default value is 1.

**Configuration Tool**
In the Configuration Tool, you can set the value of **initworker** in the **Initial number of worker threads** field in the **Resources** tab of the **Gateway daemon settings** page.

## Maximum number of worker threads

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

**maxworker=<number>**

**Description**
Set the value in the range 1-1,000,000 to specify the maximum number of worker threads. You might need to set the value of `maxworker` to less than the supported maximum value because of constraints on memory or other

system resources. You can specify that there is no limit to the number of worker threads by setting the value of **maxworker** to -1. For more information about threading limits, see "Threading model" on page 417.

You can use the **-maxworker** option to override the value of the **maxworker** parameter. For more information, for Windows see, "ctgservice command reference" on page 365, and for UNIX and Linux see, "ctgstart command reference" on page 369.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**
If this parameter is not specified, the default value is 100.

**Configuration Tool**
In the Configuration Tool, you can set the value of **maxworker** in the **Maximum number of Worker threads** field in the **Resources** tab of the **Gateway daemon settings** page. You can also set the value of **maxworker** to -1 by selecting the **Unrestricted** check box associated with the **Maximum number of Worker threads** field.

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

## Worker thread availability timeout

The **workertimeout** parameter specifies the timeout period, in milliseconds, for a worker thread to become available.

**workertimeout=<number>**

**Description**
Set the value in the range 0 - 1,000,000 to specify the time period in milliseconds. If you set the value to 0, the request is rejected unless a worker thread is immediately available.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**
If this parameter is not specified, the default value is 10,000.

**Configuration Tool**
In the Configuration Tool, you can set the value of **maxconnect** in the **Worker thread available timeout (ms)** field in the **Resources** tab of the **Gateway daemon settings** page.

## Maximum number of HTTP connections

The **maxhttpconnect** parameter defines the maximum number of HTTP clients that can be connected at the same time.

**maxhttpconnect=<number>**

**Description**
Set the value in the range 1-1,000,000 to specify the maximum number of HTTP client connections. A value of -1 indicates no limit.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**
If this parameter is not specified, the default value is 100.

**Configuration Tool**
In the Configuration Tool, you can set the value of **maxhttpconnect** in the **Maximum number of HTTP clients** field in the **Resources** tab of the **Gateway daemon settings** page. You can also set the value of **maxhttpconnect** to -1 by selecting the **Unrestricted** check box associated with the **Maximum number of HTTP clients** field. If you select the **Unrestricted** check box, no limits are applied to the number of HTTP clients.

### CICS Transaction Gateway Desktop Edition

**Maximum value**
The maximum value for this parameter is 5.

**Default value**
If this parameter is not specified, the default value is 5.

## Enable reading input from console on UNIX and Linux

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

**noinput=<on|off>**

**Description**
Set the value to **on** to disable reading of the input from the console.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**
By default this parameter is set to off. Therefore, reading of input from the console is enabled.

## Timeout for in-progress requests to complete

The **closetimeout** parameter specifies the timeout for in-progress requests to complete in milliseconds, when a Client application disconnects from the Gateway daemon.

**closetimeout=<number>**

**Description**
Set the value in the range 1-1,000,000 to specify the timeout value in milliseconds.

When a remote Client application disconnects from the CICS Transaction Gateway, the Gateway daemon might still be processing requests on behalf of that program, if the Client application disconnects before waiting for all outstanding requests to complete.

If the closetimeout parameter is set to zero, when a Client application disconnects, any outstanding work is rolled back and the connection manager and worker threads are returned to the pool. If a time greater than zero is specified, when a Client application disconnects, the Gateway daemon continues to process any outstanding requests for that client for the time specified. When the timeout expires, any outstanding work is rolled back and the connection manager and worker threads are returned to the pool.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**
If this parameter is not specified, the default value is 10,000.

**Configuration Tool**
In the Configuration Tool, you can set the value of **closetimeout** in the **Timeout for in-progress requests to complete (ms)** field in the **Resources** tab of the **Gateway daemon settings** page.

## Port for local administration

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

**adminport=<number>**

**Description**
Set the value in the range 1-65,535 to specify the port number to use for administration requests.

You can use the **-adminport** Gateway daemon override to override the value of **adminport**. On Windows platforms, the override can be set using the **ctgservice** command. On UNIX and Linux, the override can be set using the **ctgstart** command or in the ctgd configuration file. For more information, see the "ctgservice command reference" on page 365. the "ctgstart command reference" on page 369 or the "ctgd command reference" on page 364.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**
If this parameter is not specified, the default value is 2810.

**Configuration Tool**
In the Configuration Tool, you can set the value of **adminport** in the **Port for local administration** field in the **Resources** tab of the **Gateway daemon settings** page.

# 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.

Information, warning and error log messages are written to `/var/cicscli`

## Error and warning message logging

Error and warning messages can be configured by specifying the destination, log file name, the maximum number of log files, and the maximum file size of each log file.

In your configuration file, you must specify whether error and warning messages are sent to either a log file or to the console.

If you specify a log file, you must also specify the log file name, the maximum number of log files, and the maximum file size of each log file.

On UNIX and Linux, if you specify the console, you do not need to specify a log file name, maximum number of files, and maximum file size. The default behaviour of the Gateway daemon is to use a single log for informational messages and error and warning messages, the log definitions are defined with the same details.

On Windows, you can only specify that messages are sent to a log file.

## Error and warning messages destination

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

**Syntax**

**log@error.dest=<file|console>**

**Description**

Set the value to the type of destination used for error and warning messages. A destination type `file`, must be specified for Windows platforms. A destination type `file`, must be specified for UNIX and Linux platforms when the Gateway daemon is run as a background process. Otherwise, the destination type `Console` can be used to output to stderr.

**Default value on UNIX and Linux**

If this parameter is not specified, the default value is `file`. When using **ctgd** to run as a background process the log destination must be configured to use `file`.

**Default value on Windows**

If this parameter is not specified, the default value is `file`.

**Configuration Tool**

In the Configuration Tool on UNIX and Linux platforms, you can set the value of **log@error.dest** in the **Error and warning log destination** field in the **Logging** tab of the **Gateway daemon settings** page.

## Error and warning messages log file name

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

**Syntax**

**log@error.parameters=filename=<filename>**

**Description**

Set the value to the log file name for error and warning messages. 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. The directory specified in the filename must exist before the Gateway daemon is started.

**Default value**

If this parameter is not specified and the log destination is set as `file`, the default file name is `cicstg.log`. If you do not specify a path to the file, the log is created in the `<product_data_path>` directory on Windows platforms and in the `/var/cicscli` directory on UNIX and Linux. For a description of `<product_data_path>`, see Chapter 20, “File path terminology,” on page 57.

**Configuration Tool**

In the Configuration Tool on, you can set the value of

**log@error.parameters** in the **Error log file name** field in the **Logging** tab of the **Gateway daemon settings** page.

## Maximum number of error and warning message log files

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

**Syntax**

**log@error.parameters maxfiles=<number>**

**Description**

Set the value in the range 1 - 9999 to specify the maximum number of files. 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 have been created. For example, if you set the value of **log@error.parameters filename** to `ctg.log`, and **log@error.parameters maxfiles** field to 4, then the following files can 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.
```

If you set the **log@error.parameters filesize** parameter to 0, then the value of the **log@error.parameters maxfiles** parameter is ignored. If you set the**log@error.parameters filesize** parameter to greater than 0, then the value of **log@error.parameters maxfiles** must be greater than 1. If not, the error message `CTG8413E` is generated and all messages are redirected to the default log file `<product_data_path>/cicstg.log` on Windows platforms or `/var/cicscli/cicstg.log` on UNIX and Linux platforms.

**Default value**

If this parameter is not specified, the default value is 1.

**Configuration Tool**

In the Configuration Tool on, you can set the value of **log@error.parameters maxfiles** in the **Maximum number of error log files** field in the **Logging** tab of the **Gateway daemon settings** page.

## Maximum file size of error and warning messages

The **log@error.parameters filesize** parameter specifies the maximum size, in kilobytes, of the error and warning log file.

**Syntax**

**log@error.parameters filesize=<number>**

**Description**

Set the value in the range 0 - 2,097,151 to specify the maximum file size. If you set a value of 0, no limit is placed on the file size. If you set a value that is greater than 0, then you must specify that more than one log file must be kept. If not, the error message `CTG8413E` is generated and all messages are redirected to the default log file `<product_data_path>/cicstg.log` on Windows platforms or `/var/cicscli/cicstg.log` on UNIX and Linux platforms. To specify that more than one log file must be keep, set **log@error.parameters maxfiles** to a value greater than 1.

**Default value**

If this parameter is not specified, the default value is 0.

**Configuration Tool**

In the Configuration Tool on, you can set the value of **log@error.parameters filesize** in the **Error log maximum log size (KB)** field in the **Logging** tab of the **Gateway daemon settings** page.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

### Example

The following example shows the syntax required to specify that error and warning messages are sent to a log file.

```
log@error.dest=<file|console>
log@error.parameters=filename=<name>;\;
maxfiles=<number>;
filesize=<number>;
```

## Informational message logging

Informational messages can be configured by specifying the destination, log file name, the maximum number of log files, and the maximum file size of each log file.

In your configuration file, you must specify if informational messages are sent to either a log file or to the console. If you specify that your informational messages are sent to a log file, you must also specify the log file name, the maximum number of log files, and the maximum file size of each log file. On Windows, you can only specify that messages are sent to a log file. On UNIX and Linux, messages can be sent to either a log file or the console. If you specify that messages are sent to the console, a log file name, maximum number of files, and maximum file size are not required. The default behaviour of the Gateway daemon is to use a single log for informational messages and error and warning messages, the log definitions are defined with the same details.

### Informational messages destination

The **log@info.dest** parameter specifies the destination of informational messages displayed by the Gateway daemon.

**log@info.dest=<file|console>**

**Description**

Set the value to the type of destination used for informational messages. A destination type `file`, must be specified for Windows platforms. A destination type `file`, must be specified for UNIX and Linux platforms when the Gateway daemon is run as a background process. Otherwise, the destination type `Console` can be used to output to stderr.

The following example shows the syntax required to specify that informational messages are sent to a log file.

```
log@info.dest=<file|console>
log@info.parameters=filename=<name>;\
                    maxfiles=<number>;\
                    filesize=<number>;
```

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default Value on UNIX and Linux**

If this parameter is not specified, the default value is `file`. When using **ctgd** to run as a background process the log destination must be

configured to use `file`. For more information about the default name and location of the log file see "Informational messages log file name."

**Default Value on Windows**

If this parameter is not specified, the default value is `file`. For more information about the default name and location of the log file see "Informational messages log file name."

**Configuration Tool**

In the Configuration Tool on UNIX and Linux platforms, the value of **log@info.dest** is set to the same value as **log@error.dest**. You can specify a different value for **log@info.dest** by clearing the **Use the same settings as specified for Error and Warning log** check box in the **Logging** tab of the **Gateway daemon settings** page. You can then enter a different value for **log@info.dest** in the **Information log destination** field.

## Informational messages log file name

The **log@info.parameters** parameter specifies the name of the log file to be used for informational messages.

**log@info.parameters=filename=<filename>**

**Description**

Set the value to the log file name for informational messages. 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. The directory specified in the filename must exist before the Gateway daemon is started.

The following example shows the syntax required to specify that informational messages are sent to a log file.

```
log@info.dest=file
log@info.parameters=filename=<name>;\
                    maxfiles=<number>;\
                    filesize=<number>;
```

This parameter is in the "GATEWAY section of the configuration file" on page 240.

The target directory must exist before the Gateway daemon is started.

**Default value**

If this parameter is not specified and the log destination is set to `file`, the default file name is `cicstg.log`. If you do not specify a path to the file, the log is created in the `<product_data_path>` directory on Windows platforms, and in the `/var/cicscli` directory on UNIX and Linux. For a description of `<product_data_path>`, see Chapter 20, "File path terminology," on page 57.

**Configuration Tool**

In the Configuration Tool, the value of **log@info.parameters** is set to the same value as **log@error.parameters**. You can specify a different value for **log@info.parameters** by clearing the **Use the same settings as specified for Error and Warning log** check box in the **Logging** tab of the **Gateway daemon settings** page. You can then enter a different value for **log@info.parameters** in the **Information log file name** field.

## Maximum number of informational message log files

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

**log@info.parameters maxfiles=<number>**

**Description**

Set the value in the range 1 - 9999 to specify the maximum number of files. 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 **log@info.parameters maxfiles** have been created. For example, if you set the value of **log@error.parameters filename** to `ctg.log`, and **log@error.parameters maxfiles** field to 4, then the following files can 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.
```

If you set the **log@error.parameters filesize** parameter to 0, then the value of the **log@error.parameters maxfiles** parameter is ignored. If you set the**log@error.parameters filesize** parameter to greater than 0, then the value of **log@error.parameters maxfiles** must be greater than 1. If not the error message `CTG8413E` is generated and all messages are redirected to the default log file `<product_data_path>\cicstg.log` on Windows platforms or `/var/cicscli/cicstg.log` on UNIX and Linux platforms.

The following example shows the syntax required to specify that informational messages are sent to a log file.

```
log@info.dest=<file|console>
log@info.parameters=filename=<name>;\
                    maxfiles=<number>;\
                    filesize=<number>;
```

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**

If this parameter is not specified, the default value is 1.

**Configuration Tool**

In the Configuration Tool, the value of **log@info.parameters maxfiles** is set to the same value as **log@error.parameters maxfiles**. You can specify a different value for **log@info.parameters maxfiles** by clearing the **Use the same settings as specified for Error and Warning log** check box in the **Logging** tab of the **Gateway daemon settings** page. You can then enter a different value for **log@info.parameters maxfiles** in the **Maximum number of information log files** field.

## Maximum file sizes of informational messages

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

**log@info.parameters filesize=<number>**

**Description**

Set the value in the range 0 - 2,097,151 to specify the maximum file size. If you set a value of 0, no limit is placed on the file size. If you set a value that is greater than 0, then you must specify that more than one log file must be keep. If not, the error message `CTG8413E` is generated and all messages are redirected to the default log file `<product_data_path>/cicstg.log` on Windows platforms or `/var/cicscli/cicstg.log` on UNIX and Linux platforms. To specify that more than one log file must be keep, set **log@error.parameters maxfiles** to a value greater than 1.

The following example shows the syntax required to specify that informational messages are sent to a log file.

```
log@info.dest=<file|console>
log@info.parameters=filename=<name>;\
                    maxfiles=<number>;\
                    filesize=<number>;
```

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**

If this parameter is not specified, the default value is 0.

**Configuration Tool**

In the Configuration Tool, the value of **log@info.parameters filesize** is set to the same value as **log@error.parameters filesize**. You can specify a different value for **log@info.parameters filesize** by clearing the **Use the same settings as specified for Error and Warning log** check box in the **Logging** tab of the **Gateway daemon settings** page. You can then enter a different value for **log@info.parameters filesize** in the **Information log maximum log size (KB)** field.

## Log Client connections and disconnections

The **connectionlogging** parameter determines whether the CICS Transaction Gateway writes a message to the log each time that a Client application connects to or disconnects from the Gateway daemon.

**connectionlogging=<on|off>**

**Description**

Set the value to **on** to enable connection logging.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**

If this parameter is not specified, the default value is off.

**Configuration Tool**

In the Configuration Tool, you can set the value of **connectionlogging** to **on** by selecting the **Log Client connections and disconnections** check box in the **Logging** tab of the **Gateway daemon settings** page.

## Log CICS messages

The **cicslogging** parameter determines whether messages returned from CICS in IPIC error flows are logged to the CICS TG error log.

**cicslogging=<on|off>**

**Description**

Set the value to **on** to enable logging of messages returned from CICS. The messages are logged within a CICS Transaction Gateway warning message.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**

The default for this parameter is off.

**Configuration Tool**

In the Configuration Tool, you can set the value of **cicslogging** to **on** by

selecting the **Log messages received from CICS** check box in the **Logging** tab of the **Gateway daemon settings** page.

## Display TCP/IP host names

The **dnsnames** parameter defines how TCP/IP addresses are displayed in messages.

**dnsnames=<on|off>**

**Description**

Set the value to **on** to enable the display of TCP/IP addresses in messages as symbolic TCP/IP host names; these are obtained from a Domain Name System (DNS) query. This conversion makes the messages easier to read but might cause a significant reduction in performance. If the value is set to **off** TCP/IP addresses are displayed in messages in numeric form.

You can use the **ctgservice** command on Windows or **ctgstart** command on UNIX and Linux with the **-dnsnames** option to override the value of **dnsnames**.

**Note:** The **dnsnames** parameter supersedes the **nonames** parameter.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**

The default for this parameter is off.

**Configuration Tool**

In the Configuration Tool, you can set the value of **dnsnames** to **on** by selecting the **Display TCP/IP hostnames** check box in the **Logging** tab of the **Gateway daemon settings** page.

## Console output

The **quiet** parameter suppresses all console output.

**quiet=on**

**Description**

Set the **quiet** parameter to **on** to suppress console output. This parameter is available only on UNIX and Linux platforms.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**

By default, this parameter is not included in the configuration file.

**Configuration Tool**

This parameter cannot be set using the Configuration Tool.

# 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.

If you edit the configuration file using a text editor, refer to "TCP protocol parameters" on page 241

## Bind address

The **bind** parameter specifies the IP address or name of the host to which the protocol handler is bound.

**bind=<name>**

**Description**

Set the value to the IP address or name of the host. If you specify an IP address, it can be in the IPv6 format; for example, 3ffe:307:8:0:260:97ff:fe40:efab. If you specify a host name, 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.

This parameter is in the "TCP protocol parameters" on page 241 subsection of the "GATEWAY section of the configuration file" on page 240.

**Default value**

If this parameter is not specified, the default value is no IP address or name is specified.

**Configuration Tool**

In the Configuration Tool, you can set the value of **bind** in the **Bind address** field in the **TCP/IP settings** page.

## Port

The **port** parameter specifies the TCP/IP port number on which the protocol handler listens for incoming client requests.

**port=<number>**

**Description**

Set the value in the range 1 - 65,535 to specify the port number.

On Windows, you can use the **ctgservice** command with the **-port** option to override the value of the port parameter. On UNIX and Linux, you can use the **ctgstart** command with the **-port** option to override the value of the port parameter. For more information, see the Command reference.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**

This is a mandatory parameter. There is no default value.

**Configuration Tool**

In the Configuration Tool, you can set the value of **port** in the **Port** field in the **TCP/IP settings** pages.

## Connection timeout

The **connecttimeout** parameter specifies how long the protocol handler waits for a connection manager thread to become available.

**connecttimeout=<number>**

**Description**

Set the value in the range 0 - 65,536 to specify the value in milliseconds. 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 "TCP protocol parameters" on page 241 subsection of the "GATEWAY section of the configuration file" on page 240.

**Default value**
If this parameter is not specified, the default value is 2000 milliseconds.

**Configuration Tool**
In the Configuration Tool, you can set the value of **connecttimeout** in the **Connection timeout (ms)** field in the **TCP/IP settings**, **SSL settings**, or **Statistics API settings** pages.

## Idle timeout

The **idletimeout** parameter specifies the period of time, in milliseconds, that a connection is allowed to remain idle.

**idletimeout=<number>**

**Description**
Set the value in the range 0 - 9,999,999 to specify the idle timeout period in milliseconds. The idle timeout period starts after the last request has flowed down the connection. When the idle timeout has expired, the Client application is disconnected. If work is still in progress on behalf of the connection, the Client application can remain connected, depending on the setting of the "Drop working connections" on page 171 parameter. If the **idletimeout** parameter is not set or is set to zero, idle connections are not disconnected.

This parameter is in the "TCP protocol parameters" on page 241 subsection of the "GATEWAY section of the configuration file" on page 240.

**Default value**
If this parameter is not specified, the default value is 600000 milliseconds.

**Configuration Tool**
In the Configuration Tool, you can set the value of **idletimeout** in the **Idle timeout (ms)** field in the **TCP/IP settings** or **SSL settings** pages.

## Ping frequency interval

The **pingfrequency** parameter specifies the frequency that ping messages are sent by the Gateway daemon to attached Java and NET Framework-based client applications to check that the applications are still active.

**pingfrequency=<number>**

**Description**
Set the value in the range 0 - 65,536 to specify the interval period, in milliseconds, between pings. If a reply has not been received by the time the next ping message is due to be sent, the connection is disconnected. If work is still in progress on behalf of the connection, the Client application can remain connected, depending on the setting of the "Drop working connections" on page 171 parameter. If the **pingfrequency** parameter is not set or is set to zero, ping messages are not sent.

This parameter is in the "TCP protocol parameters" on page 241 subsection of the "GATEWAY section of the configuration file" on page 240.

Ping messages are not sent to C Client applications.

**Default value**

If this parameter is not specified, the default value is 60000 milliseconds.

**Configuration Tool**

In the Configuration Tool, you can set the value of **pingfrequency** in the **Ping frequency interval (ms)** field in the **TCP/IP settings** or **SSL settings** page.

## Drop working connections

The **dropworking** parameter specifies that a connection can be disconnected due to an idle timeout or a ping failure, even if work is currently in progress on behalf of this connection.

**dropworking**

**Description**

Include **dropworking** in the protocol handler parameters to specify that a connection can be disconnected. If this parameter is not included in the protocol handler parameters, connections cannot be disconnected.

This parameter is in the "TCP protocol parameters" on page 241 subsection of the "GATEWAY section of the configuration file" on page 240.

**Default value**

By default, this parameter is not included in the protocol handler parameters.

**Configuration Tool**

In the Configuration Tool, you can include **dropworking** in the protocol handler parameters by selecting the **Drop working connections** check box in the **TCP/IP settings** or **SSL settings** pages.

## SO_LINGER setting

The **solinger** parameter sets the delay, in seconds, that the Gateway waits while data is being transmitted before closing a socket, after a call has been received to close the socket.

**solinger=<number>**

**Description**

Set the value in the range 0 - 65,536 to specify the delay period in seconds. If a value greater than zero is specified and data is being transmitted when a call to close the socket is received, the Gateway waits until the data is transmitted or until the time specified before closing the socket. If a value is not specified or is set to zero, the socket is closed immediately, terminating the data transmission. However, the data transmission can be successful because TCP/IP repeats the send request for a specified period of time.

This parameter is in the "TCP protocol parameters" on page 241 subsection of the "GATEWAY section of the configuration file" on page 240.

**Default value**

If this parameter is not specified, the default value is 0.

**Configuration Tool**

In the Configuration Tool, you can set the value of **solinger** in the **SO_LINGER setting** field in the **TCP/IP settings** or **SSL settings** pages.

## Require Java Clients to use security classes

The **requiresecurity** parameter allows the Gateway to accept only connections that use security classes.

**requiresecurity**

**Description**

Include **requiresecurity** in the protocol handler parameters to specify that the CICS Transaction Gateway accepts only connections from Java Client applications that use a pair of security classes. If this parameter is not included in the protocol handler parameters, CICS Transaction Gateway accepts connections from Java Client applications that do not specify security classes.

This parameter is in the "TCP protocol parameters" on page 241 subsection of the "GATEWAY section of the configuration file" on page 240.

**Default value**

By default, this parameter is not included in the protocol handler parameters.

**Configuration Tool**

In the Configuration Tool, you can include **requiresecurity** in the configuration file by selecting the **Require Java Clients to use security classes** check box in the **TCP/IP settings** or **SSL settings** pages.

# 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.

For more information on editing the configuration file using a text editor, see "SSL protocol parameters" on page 241.

**Note:** The SSL key ring configuration parameters previously in this section, have moved to the PRODUCT section of the configuration file. For a description of the key ring file, key ring password, and key ring password encryption parameters, see "SSL key ring configuration" on page 205.

**Related information**:

"SSL key ring configuration" on page 205
To use SSL for connections between Java client applications and the Gateway daemon, or to use SSL for IPIC connections to CICS, you must configure the SSL key ring in the configuration file, `ctg.ini`.

## Bind address

The **bind** parameter specifies the IP address or name of the host to which the protocol handler is bound.

**bind=<name>**

**Description**

Set the value to the IP address or name of the host. If you specify an IP address, it can be in the IPv6 format; for example, 3ffe:307:8:0:260:97ff:fe40:efab. If you specify a host name, 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.

This parameter is in the "SSL protocol parameters" on page 241 subsection of the GATEWAY section of the configuration file.

**Default value**
If this parameter is not specified, the default value is no IP address or name is specified.

**Configuration Tool**
In the Configuration Tool, you can set the value of **bind** in the **Bind address** field in the **SSL settings** pages.

## Port

The **port** parameter specifies the TCP/IP port number on which the protocol handler listens for incoming client requests.

**port=<number>**

**Description**
Set the value in the range 1 - 65,535 to specify the port number.

On Windows, you can use the **ctgservice** command with the **-port** option to override the value of the port parameter. On UNIX and Linux, you can use the **ctgstart** command with the **-port** option to override the value of the port parameter. For more information, see the Command reference.

This parameter is in the "SSL protocol parameters" on page 241 subsection of the "GATEWAY section of the configuration file" on page 240.

**Default value**
This is a mandatory parameter. There is no default value.

**Configuration Tool**
In the Configuration Tool, you can set the value of **port** in the **Port** field in the **SSL settings** pages.

## Connection timeout

The **connecttimeout** parameter specifies how long the protocol handler waits for a connection manager thread to become available.

**connecttimeout=<number>**

**Description**
Set the value in the range 0 - 65,536 to specify the value in milliseconds. 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 "SSL protocol parameters" on page 241 subsection of the "GATEWAY section of the configuration file" on page 240.

**Default value**
If this parameter is not specified, the default value is 2000 milliseconds.

**Configuration Tool**
In the Configuration Tool, you can set the value of **connecttimeout** in the **Connection timeout (ms)** field in the **TCP/IP settings**, **SSL settings**, or **Statistics API settings** pages.

## Idle timeout

The **idletimeout** parameter specifies the period of time, in milliseconds, that a connection is allowed to remain idle.

**idletimeout=<number>**

**Description**

Set the value in the range 0 - 9,999,999 to specify the idle timeout period in milliseconds. The idle timeout period starts after the last request has flowed down the connection. When the idle timeout has expired, the Client application is disconnected. If work is still in progress on behalf of the connection, the Client application can remain connected, depending on the setting of the "Drop working connections" on page 171 parameter. If the **idletimeout** parameter is not set or is set to zero, idle connections are not disconnected.

This parameter is in the "SSL protocol parameters" on page 241 subsection of the "GATEWAY section of the configuration file" on page 240.

**Default value**

If this parameter is not specified, the default value is 600000 milliseconds.

**Configuration Tool**

In the Configuration Tool, you can set the value of **idletimeout** in the **Idle timeout (ms)** field in the **TCP/IP settings** or **SSL settings** pages.

## Ping frequency interval

The **pingfrequency** parameter specifies the frequency that ping messages are sent by the Gateway daemon to attached Java applications to check that the applications are still active.

**pingfrequency=<number>**

**Description**

Set the value in the range 0 - 65,536 to specify the interval period, in milliseconds, between pings. If a reply has not been received by the time the next ping message is due to be sent, the connection is disconnected. If work is still in progress on behalf of the connection, the Client application can remain connected, depending on the setting of the "Drop working connections" on page 171 parameter. If the **pingfrequency** parameter is not set or is set to zero, ping messages are not sent.

This parameter is in the "SSL protocol parameters" on page 241 subsection of the "GATEWAY section of the configuration file" on page 240.

**Default value**

If this parameter is not specified, the default value is 60000 milliseconds.

**Configuration Tool**

In the Configuration Tool, you can set the value of **pingfrequency** in the **Ping frequency interval (ms)** field in the **SSL settings** page.

## Drop working connections

The **dropworking** parameter specifies that a connection can be disconnected due to an idle timeout or a ping failure, even if work is currently in progress on behalf of this connection.

**dropworking**

**Description**

Include **dropworking** in the protocol handler parameters to specify that a connection can be disconnected. If this parameter is not included in the protocol handler parameters, connections cannot be disconnected.

This parameter is in the "SSL protocol parameters" on page 241 subsection of the "GATEWAY section of the configuration file" on page 240.

**Default value**

By default, this parameter is not included in the protocol handler parameters.

**Configuration Tool**

In the Configuration Tool, you can include **dropworking** in the protocol handler parameters by selecting the **Drop working connections** check box in the **TCP/IP settings** or **SSL settings** pages.

## Socket close delay

The **solinger** parameter sets the delay value in seconds for closing a socket.

**solinger**

**Description**

Set **solinger** to the number of seconds to delay closing the socket or 0 to close the socket immediately. If the setting is enabled, and data transmission is still in progress, a call to close the socket blocks the calling program. The calling program is blocked until the data is transmitted, or until the connection times out. If the setting 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.

Set the value in the range 0 through 65,536. The setting is applied to any socket used by this handler. If a value is not entered or is set to zero, the setting is disabled for any sockets used by this protocol handler.

**Default value**

There is no default value for this parameter.

**Where set**

This parameter is in the "SSL protocol parameters" on page 241 subsection of the "GATEWAY section of the configuration file" on page 240.

**Configuration Tool**

You can set the **solinger** parameter in the **SO_LINGER setting** field of the Configuration tool.

**Example**

```
solinger=10000
```

## Require Java Clients to use security classes (requiresecurity)

The requiresecurity parameter allows your Gateway to accept only connections that use security classes.

requiresecurity

**Description**

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 "SSL protocol parameters" on page 241 subsection of the "GATEWAY section of the configuration file" on page 240.

For more information, see the *CICS Transaction Gateway: Developing Applications*.

**Default value**
By default, this parameter is not included in the protocol handler parameters.

**Configuration Tool**
Check **Require Java Clients to use security classes** in the Configuration tool to allow your Gateway to accept only connections that use security classes.

## Use client authentication

The **clientauth** parameter determines if client authentication is enabled.

**clientauth=<on>**

**Description**
Include **clientauth=on** in the configuration file to specify that any client that attempts to connect using the SSL protocol handler must present its own client certificate.

This parameter is in the "SSL protocol parameters" on page 241 subsection of the "GATEWAY section of the configuration file" on page 240.

**Default value**
The default value is `off`.

**Configuration Tool**
In the Configuration Tool, you can include **clientauth=on** in the configuration file by selecting the **Use client authentication** check box in the **SSL settings** pages.

## Use only these ciphers

Use the **ciphersuites** parameter to restrict the set of cipher suites that can be used with the SSL protocol.

**ciphersuites=<name>**

**Description**
Specify the cipher suites that Java Client applications can use to connect to the CICS Transaction Gateway. You can define multiple cipher suites by separating them with a comma. If the Java Client application does not support any of the cipher suites listed, it cannot connect to the CICS Transaction Gateway. If no cipher suite is specified or the parameter is omitted, all available cipher suites can be used. Because CICS Transaction Gateway uses cipher suites provided by the Java runtime environment for the SSL protocol, the cipher suites available are dependant on the Java version. To determine which cipher suites are available for your version of Java, complete the following steps:

1. Delete the **ciphersuites** parameter from your configuration file.
2. Save the configuration file.

3. Start CICS Transaction Gateway.

   If the SSL protocol is correctly configured and CICS Transaction Gateway starts, a list of valid cipher suites is written to the Gateway daemon information log. For more information, see the documentation supplied with your Java runtime environment.

   Cipher suite information can be found in the Gateway daemon information log and Java Client application trace.

   This parameter is in the "SSL protocol parameters" on page 241 subsection of the "GATEWAY section of the configuration file" on page 240.

**Default value**

If this parameter is not specified, the default is that all available cipher suites are available.

**Configuration Tool**

In the Configuration Tool, you can set the value of **ciphersuites** in the **Use only these ciphers** field in the **SSL settings** page. Enter the cipher suite name in the field, and then click **Add** to add it to the list. To remove a cipher suite, select the suite in the list and click **Remove**.

# Chapter 39. Configuring Client daemon settings

The Client daemon settings are defined in the ctg.ini configuration file. The Client daemon manages TCP/IP and SNA connections to CICS servers.

## Maximum buffer size

The **maxbuffersize** parameter specifies the size of the transmission buffers in which application data or terminal data flows to CICS servers.

**maxbuffersize=<number>**

**Description**

Set the value in the range 4 - 32 to specify the number of kilobytes for the maximum buffer size. 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 that can be transmitted might be less than the maximum buffer size because of protocol header overheads.

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" on page 244.

**Default value**

If this parameter is not specified, the default value is 32.

**Configuration Tool**

In the Configuration Tool, you can set the value of **maxbuffersize** in the **Maximum buffer size** field on the **Resourses** tab in the **Client daemon configuration** page.

## 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.

**terminalexit=<string>**

**Description**

Set the value of a character string that is 1 - 4 characters in length. The string must not contain any blank characters and 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" on page 244.

**Default value**

The default for this parameter is EXIT.

**Configuration Tool**

In the Configuration Tool, you can set the value of **terminalexit** in the **Terminal exit** field on the **Resourses** tab in the **Client daemon configuration** page.

## Maximum servers

The **maxservers** parameter specifies the maximum number of servers that can be accessed concurrently from the Client daemon.

**maxservers=<number>**

**Description**
Set the value in the range 1 - 256 to specify the maximum number of servers.

This parameter is in the "CLIENT section of the configuration file" on page 244.

**Default value**
If this parameter is not specified, the default value is 10.

**Configuration Tool**
In the Configuration Tool, you can set the value of **maxservers** in the **Maximum servers** field on the **Resourses** tab in the **Client daemon configuration** page.

## Maximum requests

The **maxrequests** parameter specifies the maximum number of concurrent tasks that can be handled by the Client daemon.

**maxrequests=<number>**

**Description**
Set the value in the range 1 - 10,000 to specify the number of concurrent tasks. A task can be one of the following:

- A request to install a terminal emulator
- A request to install an EPI terminal
- A transaction invoked by a terminal
- An ECI unit of work
- An ESI request

The **maxrequests** parameter can be use to prevent runaway conditions, where an application can, in error, submit an excessive number of requests to a server. The actual limit might be less than the value specified if other operating system limits, such as memory constraint or communication sessions, come into effect.

This parameter is in the "CLIENT section of the configuration file" on page 244.

**Default value**
If this parameter is not specified, the default value is 256.

**Configuration Tool**
In the Configuration Tool, you can set the value of **maxrequests** in the **Maximum requests** field on the **Resourses** tab in the **Client daemon configuration** page.

**CICS Transaction Gateway Desktop Edition**
The maximum value for this parameter is 32.

## Print command

The **printcommand** parameter specifies how the operating system, that CICS Transaction Gateway is running on, must process requests to print from CICS.

**printcommand=<string>**

**Description**

Enter a string containing the instruction to the operating system on how to process print files. The string can be 1 - 256 characters in length. When a request to print is received, the Client daemon generates a temporary print file with a unique name. The temporary file name is appended to the parameter string, and the print command is run. This means that 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 processed it. This parameter overrides the value defined in the **printfile** parameter. The **cicsterm** and **cicsprnt** commands can be used to override this parameter.

For Windows, a command file might be necessary to act as an interface between the syntax of the command and more general operating system syntax. For example, if the command string is COPY printfile LPT2, a simple command file would be required to reorder the parameters. Also, you can specify only a local program in the **printcommand** parameter and UNC paths of the type `\\User1\Share\Notepad.exe` are not supported.

For UNIX and Linux, you can use a shell script with the Print command, for example, **lpr** or **rm**.

This parameter is in the "CLIENT section of the configuration file" on page 244.

**Default value**

If the **printcommand** parameter and the **printfile** parameter are omitted, the default action is to direct the print data to LPT1.

**Configuration Tool**

In the Configuration Tool, you can set the value of **printcommand** in the **Print command** field on the **Resourses** tab in the **Client daemon configuration** page.

## Print file

The **PRINTFILE** parameter identifies a file that output from print requests is directed to.

**PRINTFILE=<string>**

**Description**

Set the value to the print file name. The file name can be 1 - 256 character in length. Each print request is appended to the end of the current file. This parameter is ignored, if the **PRINTCOMMAND** parameter is specified. Also, the **cicsterm** and **cicsprnt** commands can be used to override this parameter.

This parameter is in the "CLIENT section of the configuration file" on page 244.

**Default value**

If the **PRINTCOMMAND** parameter and the **PRINTFILE** parameter are omitted, the default action is to direct the print data to LPT1.

**Configuration Tool**

In the Configuration Tool, you can set the value of **PRINTFILE** in the **Print file** field on the **Resourses** tab in the **Client daemon configuration** page.

## Code page identifier override

The **CCSID** parameter specifies the Coded Character Set Identifier (CCSID) to override the default local code page identifier used by the Client daemon.

**CCSID=<number>**

**Description**

You can set the **CCSID** parameter to override the code page identifier which the Client daemon flows to the CICS server to indicate the code page of the data being sent to the server and the code page that the server is to return data in. By default the Client daemon uses the current operating system code page. If your platform has been updated for euro support, and the CICS Server also has euro support, set the value to the correct CCSID number. 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. For support for Ja_JP locale on AIX, specify a CCSID value of 943. If you use the **CCSID** parameter to change the code page identifier, when stored data on the server is retrieved by the CICS Transaction Gateway, it might be modified if the data includes characters for which the two code pages produce different characters. `cicsterm` always displays characters based on the local code page of the workstation, despite the value specified in the **CCSID** parameter.

This parameter is in the "CLIENT section of the configuration file" on page 244.

**Default value**

By default, this parameter is not included in the configuration file.

**Configuration Tool**

In the Configuration Tool, you can set the value of **CCSID** in the **Code page identifier override** field on the **Resources** tab in the **Client daemon configuration** page.

**Related reference**:

"Use OEM code page" on page 184
The **useoemcp** parameter specifies if the CICS Transaction Gateway uses the Windows operating system OEM code page function or the Windows operating system ACP function to retrieve the code page identifier for data flowed over TCP/IP or SNA server connections.

## Server retry interval

The **srvretryinterval** parameter specifies the time between attempts made by the Client daemon to reconnect to a CICS server.

**srvretryinterval=<number>**

**Description**

Set the value in the range 0 - 3600 to specify the time interval in seconds. If

the CICS server that 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. This means that you must control the server connection using the cicscli command with the **-s=<server>** and **-x=<server>** options.

This parameter is in the "CLIENT section of the configuration file" on page 244.

**Default value**

If this parameter is not specified, the default value is 60 seconds.

**Configuration Tool**

In the Configuration Tool, you can set the value of **srvretryinterval** in the **Server retry interval (sec)** field on the **Resourses** tab in the **Client daemon configuration** page.

## Enable pop-up windows

The **enablepopups** parameter enables the display of security pop-up windows.

**enablepopups=<Y|N>**

**Description**

Set the value to Y to display security pop-up windows. Security pop-up windows are designed for use with single user installations, such as CICS Transaction Gateway Desktop Edition. Security pop-up windows cannot be enabled with Windows operating systems that provide Terminal Services or Remote Desktop Services. On these versions of Windows the **enablepopups** parameter is ignored. The IBM CICS Transaction Gateway service must also be configured to interact with the Windows desktop by completing the following steps:

1. Open the Windows Services control panel.
2. On the **Log on** tab of the properties box for the IBM CICS Transaction Gateway service, select **Allow service to interact with desktop**.
3. Ensure that the **Interactive Services Detection** service is enabled. If the **Interactive Services Detection** service is disabled and CICS TG is configured to display pop-up windows, pop-up windows are not displayed and the Client daemon becomes unresponsive.

This parameter is in the "CLIENT section of the configuration file" on page 244.

**Default value**

The default value for this parameter is N.

**Configuration Tool**

In the Configuration Tool, you can set the value of **enablepopups** to **Y** by selecting the **Enable pop-up windows** check box on the **Resourses** tab in the **Client daemon configuration** page.

## Use OEM code page

The **useoemcp** parameter specifies if the CICS Transaction Gateway uses the Windows operating system OEM code page function or the Windows operating system ACP function to retrieve the code page identifier for data flowed over TCP/IP or SNA server connections.

**useoemcp=<Y|N>**

**Description**

Set the value to Y to make CICS Transaction Gateway use the Windows operating system OEM function to retrieve the code page identifier, for example 850. If the workstation is euro-enabled, code page 850 is mapped to 858 by CICS Transaction Gateway making the euro symbol available for use by the CICS Server. If **USEOEMCP** is set to N, CICS Transaction Gateway uses the Windows operating system ACP function to get the ASCII identifier, for example 1252. The ASCII identifier can be used to provide the appropriate conversions for Latin-1, Japanese, Korean and Simplified Chinese systems. If the workstation is euro enabled, code page 1252 is mapped to code page 5348. For more information on code pages and data conversion, see Part 16, "Data conversion," on page 553.

This parameter is in the "CLIENT section of the configuration file" on page 244.

**Default value**

The default for this parameter is Y.

**Configuration Tool**

In the Configuration Tool, you can set the value of **useoemcp** to **Y** by selecting the **Use OEM code page** check box on the **Resourses** tab in the **Client daemon configuration** page.

**Related reference**:

"Code page identifier override" on page 182
The **CCSID** parameter specifies the Coded Character Set Identifier (CCSID) to override the default local code page identifier used by the Client daemon.

## Client daemon logging

You can change the location and name of the log files 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 185. The destination of informational messages can be changed to be a separate file, see "Information log file" on page 186 for more information.

The language of the log messages can be changed by running the ctgmsgs command, see "Changing the system locale" on page 96 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 184
You can change the location and name of the log files generated by the Client daemon.

## Error and warning log file

The **logfile** parameter specifies the name of the log file that is to be used for error and warning messages.

**logfile=<filename>**

**Description**

Set the value to the name of the log file that is to be used. If you do not specify a path to the file, the log is created in the `<product_data_path>` directory on Windows platforms and in the `/var/cicscli` directory on UNIX and Linux platforms.

The following table lists the code pages of the Client daemon log files for each language:

| Language | UNIX and Linux log file code page | Windows encoding |
|---|---|---|
| German (de) | 850 | UTF-8 |
| English (en) | 850 | UTF-8 |
| Spanish (es) | 850 | UTF-8 |
| French (fr) | 850 | UTF-8 |
| Italian (it) | 850 | UTF-8 |
| Japanese (ja) | 943C | UTF-8 |
| Korean (ko) | 943C | UTF-8 |
| Turkish (tr) | 857 | UTF-8 |
| Simplified Chinese (zh) | 1381 | UTF-8 |

This parameter is in the "CLIENT section of the configuration file" on page 244.

The target directory must exist before the Client daemon is started.

**Default value**

If this parameter is not specified, error and warning messages are written by default to `<product_data_path>/cicscli.log` on Windows platforms and to `/var/cicscli/cicscli.log` on UNIX and Linux platforms.

**Configuration Tool**

In the Configuration Tool, you can set the value of **logfile** in the **Error and warning log file** field on the **Logging** tab in the **Client daemon configuration** page.

## Information log file

The **logfileinfo** parameter specifies the name of the log file that is to be used for information messages.

**logfileinfo=<filename>**

**Description**

Set the value to the name of the log file that is to be used. If you do not specify a path to the file, the log is created in the `<product_data_path>` directory on Windows platforms and in the `/var/cicscli` directory on UNIX and Linux platforms.

The following table lists the code pages of the Client daemon log files for each language:

| Language | UNIX and Linux log file code page | Windows encoding |
|---|---|---|
| German (de) | 850 | UTF-8 |
| English (en) | 850 | UTF-8 |
| Spanish (es) | 850 | UTF-8 |
| French (fr) | 850 | UTF-8 |
| Italian (it) | 850 | UTF-8 |
| Japanese (ja) | 943C | UTF-8 |
| Korean (ko) | 943C | UTF-8 |
| Turkish (tr) | 857 | UTF-8 |
| Simplified Chinese (zh) | 1381 | UTF-8 |

This parameter is in the "CLIENT section of the configuration file" on page 244. The target directory must exist before the Client daemon is started.

**Default value**

If this parameter is not specified, information messages are written by default to the same file as error and warning messages, which is `<product_data_path>/cicscli.log` on Windows platforms and to `/var/cicscli/cicscli.log` on UNIX and Linux platforms.

**Configuration Tool**

In the Configuration Tool, the value of **logfileinfo** is set to the same value as **logfile**. You can specify a different value for **logfileinfo** by clearing the **Use the same file for informational messages** check box in the **Logging** tab of the **Client daemon configuration** page. You can then enter a different value for **logfileinfo** in the **Information log file** field.

## Log terminal installations and deletions

The **terminstlogging** parameter specifies if terminal installation and deletion requests are logged.

**terminstlogging=<Y|N>**

**Description**

Set the value to **Y** to enable logging of terminal installation and deletion requests.

This parameter is in the "CLIENT section of the configuration file" on page 244.

**Default value**

The default for this parameter is N.

**Configuration Tool**

In the Configuration Tool, you can set the value of **terminstlogging** to **Y** by selecting the **Log terminal installations and deletions** check box in the **Logging** tab of the **Client daemon configuration** page.

# Chapter 40. Configuring JSON web services

You can configure JSON web services on CICS Transaction Gateway to enable mobile applications to connect to CICS applications.

A JSON web service is defined by a WEBSERVICE section definition in the `ctg.ini` configuration file. It defines the URI for the web service, the name of the CICS server connection and the WSbind file to use when the web service is invoked.

In addition to a WEBSERVICE definition, CICS TG must be configured with an HTTP or HTTPS protocol handler to allow clients to connect to CICS TG to invoke the web service. For information on configuring an HTTP or HTTPS protocol handler, see "HTTP protocol settings" on page 193 and "HTTPS protocol settings" on page 194.

## Configuring a JSON web service

You define JSON web services in the WEBSERVICE section of the configuration file.

### About this task

The WEBSERVICE sections of the configuration file define the JSON web services that are exposed to HTTP and HTTPS clients.

Each web service requires a WSBind file that describes which CICS program to call when the web service is invoked and how JSON is converted to a channel or COMMAREA payload. For information on how to create and deploy a WSBind file, see *Creating a WSBind file*.

If you configure one or more web services, you must also configure a HTTP or HTTPS protocol handler to accept web service requests. For information on configuring these protocol handlers, see *HTTP protocol settings* and *HTTPS protocol settings*.

**Related information**:

"WEBSERVICE section of the configuration file" on page 247
A WEBSERVICE section in the configuration file defines a web service. A WEBSERVICE section is required for each web service exposed by CICS TG.

### Web service name

A JSON web service is defined in the SECTION WEBSERVICE of the configuration file.

**SECTION WEBSERVICE=<name>**

**Description**

Set the value to a unique name to identify the web service. The web service name can be 1-8 characters long. The following characters are valid: A - Z 0 - 9 @ # $ -

Lowercase characters, in the range a - z, are converted to uppercase.

The web service name identifies the SECTION WEBSERVICE that is exposed in the Gateway Daemon. The name is also used to enable the WSx statistics groups to have a meaningful name that can be used with admin commands and API calls.

Web service names must be unique in the configuration file. If two web services have the same name then the Gateway daemon fails to start.

**Default value**
There is no default value for this parameter.

**Configuration tool**
In the configuration tool, you can set the name of the web service in the **Web service Name** field on the **Web service** page.

## Uri

The **uri** parameter specifies a unique URI to be used by the web service.

**uri=<uri>**

**Description**
Set the value to a unique URI for the web service. If two web services specify the same URI value, then the Gateway daemon fails to start. The value specified by this property overrides a URI specified in the WSBind file.

This parameter is in the "WEBSERVICE section of the configuration file" on page 247.

The URI consists of a path and an optional query string. If the URI path ends with `*`, the web service matches any request URI that starts with the specified path. The trailing `*` character is not considered part of the URI to match. If the URI contains a query string, the web service matches requests for the URI that contains all of the specified query string parameters in any order. The following table shows some examples of the **uri** parameter and request URIs that match the parameter.

| uri parameter | Request URIs that match | Request URIs that do not match |
|---|---|---|
| MyWebService | /MyWebService<br>/MyWebService?param1=A | /MyWebService/resource |
| MyWebService/resource | /MyWebService/resource<br>/MyWebService/resource?param1=A | /MyWebService<br>/MyWebService/resource/extra |
| MyWebService/* | /MyWebService/resource1<br>/MyWebService/resource2/extra<br>/MyWebService/resource3?param1=A | /MyWebService |
| MyWebService/*?param1=A | /MyWebService/resource1?param1=A<br>/MyWebService/resource2?param1=A&param2=B | /MyWebService/resource1<br>/MyWebService/resource1?param1=B |

**Default value**

If this parameter is not specified, the Gateway daemon uses the URI specified in the WSBind file. If this parameter is not specified and the WSBind file does not specify a URI, the Gateway daemon fails to start.

## Bind file location

The **bindfile** parameter specifies the location of the WSBind file that describes which CICS program to call when the web service is invoked and how JSON is converted to a channel or COMMAREA payload.

**bindfile=<file name>**

**Description**

Set the value to the location of the WSBind file that was generated for the program that the web service will call. If you do not specify an absolute path to the file, the file is loaded from the `<product_data_path>` directory on Windows platforms and in the `/var/cicscli` directory on UNIX and Linux platforms.

This parameter is in the "WEBSERVICE section of the configuration file" on page 247.

**Default value**

There is no default value for this parameter.

**Configuration Tool**

In the Configuration Tool, you can set the value of **bindfile** in the **Bind file** field in the **Web service** page.

## Server name

The **server** specifies the name of the CICS server connection to use when the web service is started.

**server=<server name>**

**Description**

Set the value to the name of a CICS server connection to use when the web service is invoked. This can be the name of an IPICSERVER or SERVER section, or the name of a logical CICS server that will be remapped using dynamic server selection (DSS).

This parameter is in the "WEBSERVICE section of the configuration file" on page 247.

**Default value**

If this parameter is not specified, the default server is used.

**Configuration Tool**

In the Configuration Tool, you can set the value of **server** in the **Server name** field on the **Web service** page.

## Transaction identifier

The **transactionid** parameter specifies a transaction identifier that is passed to the CICS program, and optionally the transaction that is attached to run the CICS program.

**transactionid=<transaction>**

**Description**

Set the value to a transaction identifier that will be passed to the CICS program in the `EIBTRNID` field of the exec interface block (EIB). The value must be 1 - 4 characters in length. If the **defaultmirror** parameter of the web service is set to `No`, the **transactionid** value also specifies the name of the transaction which is attached when the CICS program is called. The value specified by this property overrides a transaction ID specified in the WSBind file.

This parameter is in the "WEBSERVICE section of the configuration file" on page 247.

**Default value**

If this parameter is not specified and the WSBind file specifies a transaction ID, the transaction ID specified in the WSBind file is used. This parameter must be specified or a transaction ID must be specified in the WSBind file if **defaultmirror** is set to `No`.

**Configuration Tool**

In the Configuration Tool, you can set the value of **transactionid** in the **Transaction Id** field in the **Web service** page.

## Use default mirror transaction

The **defaultmirror** parameter specifies whether the default mirror transaction should be used as the mirror transaction when the CICS program is called, or whether **transactionid** specifies the mirror transaction to use.

**defaultmirror=<Yes|No>**

**Description**

Set the value to `Yes` to attach the default mirror transaction CPMI when calling the CICS program. Set the value to `No` to use the transaction specified in the **transactionid** parameter or specified in the WSBind file as the mirror transaction.

This parameter is in the "WEBSERVICE section of the configuration file" on page 247.

**Default value**

If this parameter is not specified, the default value is `Yes`.

**Configuration Tool**

In the Configuration Tool, you can select **defaultmirror** by setting the **Use as mirror transaction** check box in the **Web service** page.

## Timeout

The **timeout** parameter specifies the number of seconds to wait for the web service request to complete.

**timeout=<value>**

**Description**

Optional. Specify a number in range 0 (no timeout) to 32767.

The number of seconds to allow for a request to complete. If a timeout value is not defined or is set to 0, then no timeout is set for the web service. Network protocol or Gateway daemon timeout settings that are set in the configuration file takes precedence.

**Default value**
If this parameter is not specified, the default value is `0`.

**Configuration Tool**
In the Configuration Tool, you can set the value of **`timeout`** in the **Request timeout** field in the **Web service** page

## HTTP protocol settings

You define the parameters for the HTTP protocol in the GATEWAY section of the configuration file.

You can use the CICS Transaction Gateway configuration tool to configure the HTTP protocol settings, or edit the parameters in the configuration file directly. If you edit the configuration file using a text editor, refer to *HTTP protocol parameters*.

### Bind address

The **`bind`** parameter specifies the IP address or name of the host to which the protocol handler is bound.

**`bind=<name>`**

**Description**
Set the value to the IP address or name of the host. If you specify an IP address, it can be in the IPv6 format; for example, 3ffe:307:8:0:260:97ff:fe40:efab. If you specify a host name, it is resolved on startup.

This parameter is in the "HTTP protocol parameters" on page 242 subsection of the "GATEWAY section of the configuration file" on page 240.

**Default value**
If the bind parameter is not specified or is blank, the default behavior is to bind to all IP addresses.

**Configuration Tool**
In the Configuration Tool, you can set the value of **`bind`** in the **Bind address** field in the **HTTP settings** page.

### Port

The **`port`** parameter specifies the TCP/IP port number on which the protocol handler listens for incoming client requests.

**`port=<number>`**

**Description**
Set the value in the range 1 - 65,535 to specify the port number.

On Windows, you can use the **`ctgservice`** command with the `-httpport` option to override the value of the port parameter. On UNIX and Linux, you can use the **`ctgstart`** command with the `-httpport` option to override the value of the port parameter. For more information, see the Command reference.

This parameter is in the "HTTP protocol parameters" on page 242 subsection of the "GATEWAY section of the configuration file" on page 240.

**Default value**
This is a mandatory parameter. There is no default value.

**Configuration Tool**
In the Configuration Tool, you can set the value of **port** in the **Port** field in the **HTTP settings** pages.

# HTTPS protocol settings

You define the parameters for the HTTPS protocol in the GATEWAY section of the configuration file.

You can use the CICS Transaction Gateway configuration tool to configure the HTTPS protocol settings, or edit the parameters in the configuration file directly. If you edit the configuration file using a text editor, refer to *HTTPS protocol parameters*.

## Bind address

The **bind** parameter specifies the IP address or name of the host to which the protocol handler is bound.

**bind=<name>**

**Description**
Set the value to the IP address or name of the host. If you specify an IP address, it can be in the IPv6 format; for example, 3ffe:307:8:0:260:97ff:fe40:efab. If you specify a host name, it is resolved on startup.

This parameter is in the "HTTPS protocol parameters" on page 243 subsection of the "GATEWAY section of the configuration file" on page 240.

**Default value**
If the bind parameter is not specified or is blank, the default behavior is to bind to all IP addresses.

**Configuration Tool**
In the Configuration Tool, you can set the value of **bind** in the **Bind address** field in the **HTTPS settings** pages.

## Port

The **port** parameter specifies the TCP/IP port number on which the protocol handler listens for incoming client requests.

**port=<number>**

**Description**
Set the value in the range 1 - 65,535 to specify the port number.

On Windows, you can use the **ctgservice** command with the **-httpsport** option to override the value of the port parameter. On UNIX and Linux, you can use the **ctgstart** command with the **-httpsport** option to override the value of the port parameter. For more information, see the Command reference.

This parameter is in the "HTTPS protocol parameters" on page 243 subsection of the "GATEWAY section of the configuration file" on page 240.

**Default value**
This is a mandatory parameter. There is no default value.

**Configuration Tool**
In the Configuration Tool, you can set the value of **port** in the **Port** field in the **HTTPS settings** pages.

## Use client authentication

The **clientauth** parameter determines if client authentication is enabled.

**clientauth=<on>**

**Description**
Include **clientauth=on** in the configuration file to specify that any client that attempts to connect using the SSL protocol handler must present its own client certificate.

This parameter is in the "HTTPS protocol parameters" on page 243 subsection of the "GATEWAY section of the configuration file" on page 240.

**Default value**
By default, client authentication is not enabled.

**Configuration Tool**
In the Configuration Tool, you can include **clientauth=on** in the configuration file by selecting the **Use client authentication** check box in the **HTTPS settings** pages.

## Use only these ciphers

Use the **ciphersuites** parameter to restrict the set of cipher suites that can be used with the HTTPS protocol.

**ciphersuites=<name>**

**Description**
Specify the cipher suites that Java Client applications can use to connect to the CICS Transaction Gateway. You can define multiple cipher suites by separating them with a comma. If the Java Client application does not support any of the cipher suites listed, it cannot connect to the CICS Transaction Gateway. If no cipher suite is specified or the parameter is omitted, all available cipher suites can be used. Because CICS Transaction Gateway uses cipher suites provided by the Java runtime environment for the HTTPS protocol, the cipher suites available are dependant on the Java version. To determine which cipher suites are available for your version of Java, complete the following steps:

1. Delete the **ciphersuites** parameter from your configuration file.
2. Save the configuration file.
3. Start CICS Transaction Gateway.

If the HTTPS protocol is correctly configured and CICS Transaction Gateway starts, a list of valid cipher suites is written to the Gateway daemon information log. For more information, see the documentation supplied with your Java runtime environment.

Cipher suite information can be found in the Gateway daemon information log and Java Client application trace.

This parameter is in the "HTTPS protocol parameters" on page 243 subsection of the "GATEWAY section of the configuration file" on page 240.

**Default value**

If this parameter is not specified, the default is that all available cipher suites are available.

**Configuration Tool**

In the Configuration Tool, you can set the value of **ciphersuites** in the **Use only these ciphers** field in the HTTPS settings page. Enter the cipher suite name in the field, and then click **Add** to add it to the list. To remove a cipher suite, select the suite in the list and click **Remove**.

# Chapter 41. 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.

## Creating and maintaining digital certificates

Digital certificates are used for identifying either end of 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. iKeyman and iKeytool are shipped in both the JRE and Java SDK packages.

**Related information**:

"Using keytool for certificate management" on page 201
The keytool command line application is provided with the Java SDK.

## 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 201.

## 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 `MyServerkeyring.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 `MyServerkeyring.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 `MyClientkeyring.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 **Signer Certificates**.
2. Select **Add**.
3. Locate the stored Server Base64-encoded ASCII certificate file. In our example, this is `exampleServercert.arm`.
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.

## 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 complete 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.

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 is provided with the Java SDK.

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: 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*

The abbreviations in the distinguished name have the following meaning:

- cn = common name
- o = organization
- ou = organization unit
- l = city/locality
- s = state/province
- c = country 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=someserver.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 certificate

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

Follow these steps to configure your SSL clients.

If your server does not use client authentication you need only complete the first task, Create a client key ring and import the server's signer certificate.

### 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: 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*

The abbreviations in the distinguished name have the following meaning:

- cn = common name
- o = organization
- ou = organization unit
- l = city/locality
- s = state/province
- c = country 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.

## SSL key ring configuration

To use SSL for connections between Java client applications and the Gateway daemon, or to use SSL for IPIC connections to CICS, you must configure the SSL key ring in the configuration file, `ctg.ini`.

## Key ring file

The **keyring** parameter specifies the name of the key ring.

**keyring=<file>**

**Description**

Set the value to the name of a keyring, that can be stored in an external security manager, that the protocol uses. Specify 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.The user ID that the Gateway daemon is running under must be able to access the key ring file. If you are using a RACF® keyring file, the value of the **keyring** parameter is the name of the RACF key ring.

You can use the **ctgservice -R -A-keyring=***file* on Windows or **ctgstart -keyring=***file* on UNIX and Linux command to override the value of **keyring**.

This parameter is in the PRODUCT section of the configuration file.

**Default value**

There is no default value.

**Configuration Tool**

In the Configuration Tool, you can set the value of **keyring** in the **Key ring file** field in the **CICS Transaction Gateway page** page.

## Key ring password

The **keyringpw** parameter specifies the password associated with the key ring file defined in the **keyring** parameter.

**keyringpw=<password>**

**Description**

Set the value to the password for the keyring file. If you are using a RACF keyring, do not specify the **keyringpw** parameter.

You can use the command **ctgservice -R -A-keyringpw=***password* on Windows or **ctgstart -keyringpw=***password* on UNIX and Linux , to override the value of **keyringpw**.

This parameter is in the PRODUCT section of the configuration file.

**Default value**

There is no default value. This parameter must be specified if the keyring parameter is specified.

**Configuration Tool**

In the Configuration Tool, you can set the value of **keyringpw** in the **Key ring password** field in the **CICS Transaction Gateway** page. The **keyringpwscrambled** parameter is automatically set to on. Therefore, in the configuration file, the password is displayed in a form that prevents an observer from easily reading it.

## Key ring password encryption

The **keyringpwscrambled** parameter specifies whether the **keyringpw** parameter value was encrypted by the Configuration Tool.

**keyringpwscrambled=<on|off>**

**Description**
The Configuration Tool sets this value to **on** when it encrypts the value of **keyringpw**.

This parameter is in the "PRODUCT section of the configuration file" on page 239.

**Default value**
If this parameter is not specified, the default value is off.

**Configuration Tool**
This parameter is always set to on by the Configuration Tool.

## SSL configuration for IPIC connections

SSL can be defined for local or remote IPIC connections.

### Local mode

In local mode, IPIC connections use the SSL key ring settings of either the Java base class or the resource adapter.

1. To configure SSL for the Java base classes:
   a. Create a java.util.Properties object.
   b. Add the following properties:
      1) **JavaGateway.SSL_KEYRING_CLASS, <keyring file location>**
      2) **JavaGateway.SSL_KEYRING_PASSWORD, <password>**
   c. Set the properties on the JavaGateway by calling the setProtocolProperties() method, passing the java.util.Properties object.
   d. Define the server name as `ssl://<server_name>:<port>`. Set the server name on the ECIRequest object and not on the JavaGateway object.
2. To configure an SSL connection for a resource adapter, edit the connection factory custom properties:
   a. Set the **serverName** property to `ssl://<server_name>:<port>`.
   b. Set the **keyRingClass** property to the location of the key ring file or Java keystore.
   c. If the **keyRingClass** property specifies a Java keystore, then set the **keyRingPassword** property to the password of the key ring file.

### Remote mode

To configure the Gateway daemon to use SSL connections to CICS:

1. Set the key ring parameters for the Gateway daemon. For more information, see "SSL key ring configuration" on page 205.
2. To enable SSL on each IPIC connection, set the **ssl** parameter in the "IPICSERVER section of the configuration file" on page 244 to Y.
3. If you want to limit the cipher suites that are enabled for the connection, set the **ciphersuites** parameter to a comma separated list of cipher suites to use.

## SP800-131A compliance

SP800-131A compliance strengthens security by requiring the use of stronger cryptographic keys and more robust algorithms.

To specify that SP800-131A transition or strict compliance is required, set the Java system property `com.ibm.jsse2.sp800-131` as follows:

```
com.ibm.jsse2.sp800-131=<transition|strict|off>
```

Set the property for the Java client application in local mode and the Gateway daemon in remote mode. For strict support on an SSL connection between a Java client application and the Gateway daemon, both the Java client application and Gateway daemon must specify `com.ibm.jsse2.sp800-131=strict`.

Additionally, for strict support with NET Framework-based clients, the **SslGatewayConnection** property must be configured to use TLS 1.2. This property can be set with the **EnabledSslProtocols** property or **CtgSslProtocols** application configuration setting.

If using Cipher suites that use AES_256 then the Gateway daemon JVM must be updated with the Unrestricted JCE policy files placed in the `<install_path>/jvm170/lib/security/` directory. To obtain the Unrestricted JCE policy files and for more information, see IBM SDK Policy Files.

CICS Transaction Gateway supports SP800-131a strict mode on IPIC SSL connections in local and remote mode to CICS Transaction Server and IBM TXSeries versions which also support SP800-131a strict mode. This includes support for requests from IBM WebSphere Application Server using the CICS ECI resource adapter.

For more information, see the National Institute of Standards and Technology (NIST) Special Publications 800-131a at http://csrc.nist.gov/publications/nistpubs/800-131A/SP800-131A.pdf.

## FIPS 140-2 compliance

FIPS 140-2 is a standard which certifies cryptographic modules used by applications.

CICS Transaction Gateway can be configured to use FIPS 140-2 certified modules to maintain the confidentiality and integrity of the information protected by the modules.

To configure the CICS TG to use FIPS 140-2 certified modules:

1. Locate the `java.security` file in the `<install_dir>/jvm170/lib/security` directory.
2. Locate the providers section in the file and add the following two lines at the top:

   ```
   security.provider.1=com.ibm.fips.jsse.IBMJSSEFIPSProvider
   security.provider.2=com.ibm.fips.crypto.fips.provider.IBMJCEFIPS
   ```

   Renumber the existing providers as required.
3. Save the file and restart the CICS Transaction Gateway.

# Chapter 42. Configuring identity propagation

Identity propagation configuration tasks are required on RACF, CICS Transaction Server and IBM WebSphere Application Server. Identity propagation must also be activated in CICS Transaction Gateway.

## 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 documentation.

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*IBM z/OS Security Server RACF Command Language Reference*.

**Note:** It is not possible to modify a distributed identity filter.

## Configuring identity propagation on CICS Transaction Server

The steps required to configure identity propagation on CICS Transaction Server.

CICS Transaction Server requires the following:

- The IBM z/OS identity propagation function provided in IBM z/OS, Version 1.11 or later.
- CICS Transaction Server for IBM z/OS Version 4.1 or later with the APAR fixes described in “Configurations that support identity propagation” on page 407. To download these fixes, go to Fix list for CICS Transaction Server for IBM 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 393.

# Configuring identity propagation on IBM WebSphere Application Server

Configuration is required on IBM WebSphere Application Server to enable identity propagation.

## Setting up the identity propagation login module

IBM WebSphere Application Server must be configured to specify a user registry to enable user ID and password verification for applications. Any registry supported by IBM WebSphere Application Server is supported by CICS Transaction Gateway. Examples of the registries supported by IBM 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 IBM WebSphere Application Server documentation.

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 IBM 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 IBM WebSphere Application Server, see the IBM WebSphere Application Server documentation.

### 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 to CICS, you set the `propIdentity` custom property on the CICS Transaction Gateway identity propagation login module. You do this from the IBM 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 IBM WebSphere administrative console; click **Security** > **Global security**, expand **Java Authentication and Authorization Service** and select **Application logins**. In the new window, click **New**.
2. Enter `CTG_idprop` as the Alias.
3. Click **New** under JAAS login modules.
4. Enter `com.ibm.ctg.security.idprop.LoginModule` as the Module class name.
5. Clear the **Use login module proxy** check box.
6. Select REQUIRED from the **Authentication strategy** drop-down list.
7. Under "Custom properties" create an entry with Name as `propIdentity` and Value as `RunAs`.
8. 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 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 IBM WebSphere Application Server. For more information, see Chapter 93, "Deploying a CICS resource adapter," on page 451.
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 IBM WebSphere Application Server. For more information, see “Configuring identity propagation on IBM WebSphere Application Server” on page 210.
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 IBM WebSphere Application Server, see the IBM WebSphere Application Server documentation.

# Chapter 43. Configuring high availability

High availability is supported by the default server, the CICS request exit, and the Windows Workload Manager.

## Default server

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

**defaultserver=<name>**

**Description**
Specify a CICS server name that is used CICS Transaction Gateway for Client application 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.

**Default value**
There is no default value for this parameter.

**Configuration Tool**
In the Configuration Tool, you can set the value of **defaultserver** from the **Default Server** field on the **CICS Transaction Gateway** page.

## CICS request exit

The **cicsrequestexit** parameter specifies the class used to perform dynamic CICS server selection for ECI requests and ESI requests.

**cicsrequestexit=<class>**

**Description**
Set the value to a fully qualified class that implements the com.ibm.ctg.ha.CICSRequestExit interface. The class must be on the class path of the Gateway daemon. The Gateway daemon supports only a single exit at any time, for more information see Chapter 91, "CICS request exit," on page 441.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**
There is no default value.

**Configuration Tool**
In the Configuration Tool, you can set the value of **cicsrequestexit** in the **CICS Request Exit** field in the **Resources** tab of the **Gateway daemon settings** page.

## Configuring the Windows Workload Manager

Use the CICS Transaction Gateway Configuration Tool to configure the Workload Manager settings. Alternatively, you can edit the LOADMANAGER and SERVER sections of the configuration file directly.

To display the Workload Manager settings panel in the Configuration Tool, select the Workload Manager node in the navigation panel. Most settings map to the parameters in the LOADMANAGER section of the ctg.ini file, although the Workload Manager also uses the **region** parameter in the SERVER section.

Read Chapter 92, "Windows Workload Manager," on page 443, before configuring the Workload Manager. This section explains important concepts, such as server groups, and the behavior of the workload balancing algorithms.

The Workload Manager is used exclusively on Windows platforms for connections to CICS servers over TCP/IP or SNA.

## Enable the Workload Manager

The **LOADMANAGER** parameter defines whether the Workload Manager is enabled or not.

**LOADMANAGER=<Y|N>**

**Description**
Set the value to Y to use the Workload Manager to balance the workload in your system effectively across CICS servers connected by TCP/IP and SNA.

This parameter is in the "CLIENT section of the configuration file" on page 244.

**Default value**
The default for this parameter is N.

**Configuration Tool**
In the Configuration Tool, you can set the value of **LOADMANAGER** to **Y** by selecting the **Enable the Workload Manager** check box in the **Workload Manager settings** page.

## Configuring a server group

The **REGION** parameter is used to assign a server to a Workload Manager server group.

**REGION=<name>**

**Description**
Set the value to the name of the Workload Manager server group. This group can be disabled using the **NOBALANCE** parameter, for more information see, "Disabling workload balancing for a program" on page 217.

The Workload Manager is used exclusively on Windows platforms for connections to CICS servers over TCP/IP or SNA. For more information, see Chapter 92, "Windows Workload Manager," on page 443.
This parameter is in the "SERVER section of the configuration file" on page 245.

**Default value**
There is no default value for this parameter.

**Configuration Tool**
In the Configuration Tool, you can set the value of **REGION** in the **Server**

**group definition** page. This page can be accessed by selecting an existing server in the **Server Groups** section of the **CICS Transaction Gateway** pane or by creating a server group.

To create a server group you can click **Options** > **New Server Group**, click the **New Server Group** icon, or click the **New Server Group** button on the **Server Groups** page. You are then prompted to enter a name for the server group. This name must be 1 - 8 characters in length.

To add servers to the group, select the server in the **Servers not in server group** list and click the **<<** button. The server is added to the **Servers in server group**. To remove a server from the group, select the server in the **Servers in server group** list and click the **>>** button. The server is added to the **Servers not in server group**.

The entry, **REGION**=*Server Group Name*, is added to the **SECTION SERVER** sections of the configuration file for each of the servers listed in the **Servers in server group**, where *Server Group Name* is the name specified in the **Server group name** field in the **Server group definition** page.

## Round robin or biasing

In the configuration file set the **TYPE** parameter value to 1 for the biased algorithm, or 0 for the round robin algorithm. The biased algorithm is selected by default.

**TYPE=<1|0>**

**Description**

Set the value to 1 to specify the biased algorithm. Set the value to 0 to specify the round robin algorithm. For more information about biased and round robin algorithms, see "Load balancing algorithms" on page 445.

The Workload Manager is used exclusively on Windows platforms for connections to CICS servers over TCP/IP or SNA. For more information, see Chapter 92, "Windows Workload Manager," on page 443.
This parameter is in the "LOADMANAGER section of the configuration file" on page 247.

**Default value**

This is a required parameter, there is no default value. If it not specified, the Windows Workload Manager is disabled.

**Configuration Tool**

In the Configuration Tool, you can set the value of **TYPE** by selecting either of the **Round robin** or **Biased** radio buttons in the **Workload Manager settings** page.

## Configuring workload biasing

The **bias** parameter defines how the Workload Manager distributes program requests to server groups or individual servers.

**BIAS=<string>**

**Description**

Define a program name; the server group or server that the program requests can be sent to; and the bias value. A bias parameter is required for each program that is to be workload balanced and defines how program requests are distributed between server groups. The value of the string must match the following format:

```
<program>;<server group 1>:<bias value>;<server>:<bias value>;
<server group 2>:<bias value>;...
```

If you are using the Round robin algorithm, specify a bias value of 0. For more information about bias values, see "The biasing algorithm" on page 446. If a server is not in a server group, you can specify the server name in place of a server group name, for example:

```
BIAS=program1;servergroup1:1;servergroup2:3;
BIAS=program2;servergroup1:1;servergroup3:1;servergroup4:1;
```

This parameter is in the "LOADMANAGER section of the configuration file" on page 247.

**Configuration Tool**

In the Configuration Tool, you can set the value of **BIAS** in the **Program definition for managing workload** page. This page can be accessed by selecting an existing program in the **Programs** section of the **CICS Transaction Gateway** pane or by creating a program definition.

To create a program definition you can click **Options** > **New Program Definition**, click the **New Program Definition** icon, or click the **New Program Definition** button on the **Server Groups** page. You are then prompted to enter a program name. This name must be 1 - 8 characters in length.

To add server groups or servers to the list of servers used for biasing, select the server group or server in the **Servers not used for biasing** list and click **<<**. The server group or server is added to the **Servers used for biasing**. To remove a server group or server from the group, select the server group or server in the **Servers used for biasing** list and click the **>>** button. The server group or server is added to the **Servers not used for biasing**.

Set the bias value by clicking the **Bias value** field for a particular server, and then enter a number. The ratio field automatically updates. This field describes the ratio of the distribution of program requests between the servers in the **Servers used for biasing** list.

## Server group timeout

The **timeout** parameter specifies the delay, in seconds, after which an invalid server can again be included in the Workload Manager server group.

**timeout=<number>**

**Description**

Set the value in the range 0 - 3600 to specify the delay in seconds. When a server is unavailable, the server is flagged as invalid, and is no longer included in the workload balancing algorithm used by the Workload Manager.

The Workload Manager is used exclusively on Windows platforms for connections to CICS servers over TCP/IP or SNA. For more information, see Chapter 92, "Windows Workload Manager," on page 443.
This parameter is in the "LOADMANAGER section of the configuration file" on page 247.

**Default value**

This is a required parameter, there is no default value. If it not specified, the Windows Workload Manager is disabled.

**Configuration Tool**

In the Configuration Tool, you can set the value of **timeout** in the **Server group timeout** field in the **Workload Manager settings** page.

## Disabling workload balancing for a program

The **NOBALANCE** parameter disables workload balancing for the program defined.

**NOBALANCE=<string>**

**Description**

Specify a program for which the Workload Manager does not balance program requests. The string specified must also include the associated server groups or servers that the program requests can be sent to and the bias values for these server groups or servers. The value of the string must match the following format:

```
<program>;<server group 1>:<bias value>;<server>:<bias value>;
<server group 2>:<bias value>;...
```

The Workload Manager is used exclusively on Windows platforms for connections to CICS servers over TCP/IP or SNA. For more information, see Chapter 92, "Windows Workload Manager," on page 443.

This parameter is in the "LOADMANAGER section of the configuration file" on page 247.

**Default value**

There is no default value for this parameter.

**Configuration Tool**

In the Configuration Tool, you can specify that program requests for an individual program are not balanced by the Workload Manager, by clearing the **Manage workload for this program** check box in the **Program definition for managing workload** page. This page can be accessed by selecting an existing program in the **Programs** section of the **CICS Transaction Gateway** pane. The **BIAS** parameter for the program in the **SECTION LOADMANAGER** section of the configuration file is replaced by a **NOBALANCE** parameter has the same string that was specified in the **BIAS** parameter. If you reselect the **Manage workload for this program** check box, the original **BIAS** parameter is restored.

## Disabling workload balancing for a server group

The **NOTMANAGED** parameter disables workload balancing for the server group defined.

**NOTMANAGED=<string>**

**Description**

Specify one or more server groups for which the Workload Manager does not balance requests. For more than one server group, separate each server group with a semicolon. If you disable workload balancing for a server group, the Workload Manager connects only to the server specified in the call. The Workload Manager does not re-send failed requests to a server that has workload balancing disabled.

To specify that all load balancing is enabled for all server groups, do not specify the **NOTMANAGED=<string>** in your configuration file. If load balancing is enabled for a server group, the first time an instance of the

Client daemon makes a request to this server group, it selects a CICS server from the server group at random. For the duration of the life of the Client daemon instance, all subsequent requests to that server group are sent to the same CICS server. All servers not assigned to a server group are also considered for workload management, and a server not included in a server group is thought to be in a server group of its own.
This parameter is in the "LOADMANAGER section of the configuration file" on page 247.

**Default value**

There is no default value for this parameter.

**Configuration Tool**

In the Configuration Tool, you can specify that requests to a server group are not balanced by the Workload Manager, by clearing the **Manage workload for this server group** check box in the **Server group definition** page. This page can be accessed by selecting an existing server group in the **Client daemon** section. Reselecting the **Manage workload for this server group** check box re-enables workload balancing for this server group.

# Chapter 44. Configuring monitoring and statistics

You can configure CICS Transaction Gateway to record monitoring and statistics data.

## Configuring request monitoring exits for the Gateway daemon

The **requestexits** parameter specifies a list of one or more classes that perform request monitoring.

**requestexits=<fully_qualified_class_name1[,fully_qualified_class_name_n]>**

**Description**

Specify the fully qualified class names for a request monitor classes. You can define multiple classes by separating them with a comma. For example:

```
requestexits=com.ibm.ctg.samples.requestexit.1stMonitor,com.ibm.ctg.
samples.requestexit.2ndMonitor
```

You can use the **-requestexits=***exits* option on the **ctgservice** command on Windows or the **ctgstart** command on UNIX and Linux to override the value of **requestexits**. For more information, see Command reference.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**

There is no default value for this parameter.

**Configuration Tool**

In the Configuration Tool, you can set the value of **requestexits** using the **Use these request monitors** field in the **Monitoring** tab of the **Gateway daemon settings** page. Enter the fully qualified class name for a request monitor class in the field, and then click **Add** to add the class name to the list. To remove a class, select the class name in the list and click **Remove**.

**Note:** Statistics and Monitoring exits are not guaranteed to be accurate when a CICS Intercept plug-in is enabled. The CICS Request exit will not be called when a CICS Intercept plug-in does not allow a request to continue. For more information, see Chapter 35, "Configuring for application testing," on page 105.

## Configuring request monitoring for the Gateway classes

The configuration of the request monitoring exits 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 7. 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 451
The available deployment parameters for the ECI resource adapter 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.Gateway daemon monitoring resources.

## 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.

If you edit the configuration file with a text editor, refer to "Statistics API protocol parameters" on page 243.

### Bind address

The **bind** parameter specifies the IP address or name of the host to which the protocol handler is bound.

**bind=<name>**

**Description**

Set the value to the IP address or name of the host. If you specify an IP address, it can be in the IPv6 format; for example, 3ffe:307:8:0:260:97ff:fe40:efab. If you specify a host name, 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.

This parameter is in the "Statistics API protocol parameters" on page 243 subsection of the GATEWAY section of the configuration file.

**Default value**

If this parameter is not specified, the default value is no IP address or name is specified.

**Configuration Tool**

In the Configuration Tool, you can set the value of **bind** in the **Bind address** field in the **Statistics API settings** page.

## Port

The **port** parameter specifies the TCP/IP port number on which the protocol handler listens for incoming client requests.

**port=<number>**

**Description**

Set the value in the range 1 - 65,535 to specify the port numbers.

On Windows, you can use the **ctgservice** command with the **-port** option to override the value of the port parameter. On UNIX and Linux, you can use the **ctgstart** command with the **-port** option to override the value of the port parameter. For more information, see the Command reference.

This parameter is in the "Statistics API protocol parameters" on page 243 subsection of the GATEWAY section of the configuration file.

**Default value**

If this parameter is not specified, the default value is 2980.

**Configuration Tool**

In the Configuration Tool, you can set the value of **port** in the **Port** field in the **Statistics API settings** pages.

## Connection timeout

The **connecttimeout** parameter specifies how long the protocol handler waits for a connection manager thread to become available.

**connecttimeout=<number>**

**Description**

Set the value in the range 0 - 65,536 to specify the value in milliseconds. 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" on page 240.

**Default value**

If this parameter is not specified, the default value is 2000 milliseconds.

**Configuration Tool**

In the Configuration Tool, you can set the value of **connecttimeout** in the **Connection timeout (ms)** field in the **TCP/IP settings**, **SSL settings**, or **Statistics API settings** pages.

### Maximum number of connections

The **maxconn** parameter specifies the maximum number of applications that can simultaneously connect to Gateway daemon to perform statistics queries.

**maxconn=<number>**

**Description**

Set the value to the maximum number of connections.

This parameter is in the "Statistics API protocol parameters" on page 243 subsection of the "GATEWAY section of the configuration file" on page 240.

**Default value**

If this parameter is not specified, the default value is 5.

**Configuration Tool**

In the Configuration Tool, you can set the value of **maxconn** in the **Maximum connections** field in the **Statistics API settings** page.

## Statistics interval

The **statint** parameter specifies the recording interval for system statistics.

**statint=<HHMMSS>**

**Description**

Set the value in the range 000100 - 240000 to define the CICS Transaction Gateway statistics recording interval duration. The value must be in the format HHMMSS. The hours, HH, must be specified in the range 0 - 24. The minutes, MM, and seconds, SS, must be specified in the range 00 - 59. If the value set is less than the minimum it is changed to 000100. If the value set is greater than the maximum, it is changed to 240000. If the value set does not have the correct format, it is changed to the default value.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**

If this parameter is not specified, the default value is 030000.

**Configuration Tool**

In the Configuration Tool, you can set the value of **statint** in the **Statistics interval (HHMMSS)** field in the **Monitoring** tab of the **Gateway daemon settings** page.

## Statistics end of day time

The **stateod** parameter specifies the end of day time.

**stateod=<HHMMSS>**

**Description**

Set the value in the range 000000 - 235959 to define the CICS Transaction Gateway end of day time in local time. The value must be in the format HHMMSS, where midnight is 000000 and one second before midnight is 235959. The hours, HH, must be specified in the range 00 - 23. The minutes, MM, and seconds, SS, must be specified in the range 00 - 59. For example, to specify an end of day time of 3 mins and 9 seconds after midnight, the value of the **stateod** parameter must be set to 000309. If the value set is greater than the maximum, it is changed to 235959. If the value

set does not have the correct format, it is changed to the default value. The end of day time is used as a point of reference for the clock rather than to the CICS CICS Transaction Gateway startup time. This also determines the point at which statistics are reset and potentially recorded.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**
If this parameter is not specified, the default value is 0.

**Configuration Tool**
In the Configuration Tool, you can set the value of **stateod** in the **Statistics End of Day time (HHMMSS)** field in the **Monitoring** tab of the **Gateway daemon settings** page.

## Configuring statistics recording to a file

You can configure CICS Transaction Gateway to record statistics data to an XML file.

To do this, you must enable statistics recording using the **statsrecording** parameter; specify that statistics are written to an XML statistics log file using the **log@statistics.dest** parameter; and define the file name of the XML statistics log file using the **log@statistics.parameters** parameter. The interval that statistics are recorded is defined by the statint and stateod parameters.

### Enable statistic recording to an XML file

The **statsrecording** parameter determines whether statistics are written to an XML statistic log file.

**statsrecording=<on|off>**

**Description**
Set the value to **on** to enable recording of statistics.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**
The default for this parameter is off.

**Configuration Tool**
In the Configuration Tool, you can set the value of **statsrecording** to **on** by selecting the **Enable statistics recording** check box in the **Monitoring** tab of the **Gateway daemon settings** page.

### Statistics log destination

The **log@statistics.dest** parameter specifies that statistics are written to an XML statistics log file.

**log@statistics.dest=file**

**Description**
Set the value `file` to specify that statistics are written to an XML statistics log file. If **statsrecording** is set to `on` and **log@statistics.dest=file** is not specified in the configuration file, the CICS Transaction Gateway fails to start.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**
There is no default value for this parameter.

**Configuration Tool**
In the Configuration Tool, if you have set the value of **statsrecording** to **on** by selecting the **Enable statistics recording** check box in the **Monitoring** tab of the **Gateway daemon settings** page, then the Configuration Tool automatically adds **log@statistics.dest=file** to the configuration file.

## Statistics log file name

The **log@statistics.parameters** parameter specifies the name of the XML statistics log file used for statistics recording.

**log@statistics.parameters=filename=<filename>**

**Description**
Set the value to the statistics log file name. 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. The directory specified in **filename** must exist before the Gateway daemon is started. When a file name is defined without a directory, the statistics log file is created in the <product_data_path> directory for Windows platforms and the /var/cicscli directory for UNIX and Linux platforms.

To ensure that each Gateway daemon instance has a unique statistics log file, a time stamp that specifies when the Gateway daemon started is added to the file name. If the file name specified has a file extension, the time stamp is inserted between the file name and the file extension, otherwise the time stamp is appended to the end of the file name. The time stamp has the following format _YYYYMMDD-hhmmss, where YYYY is the year, MM is the month, DD is the day, hh is the hour, mm is the minute, and ss is the second that the Gateway daemon was started.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**
There is no default value for this parameter.

**Configuration Tool**
In the Configuration Tool, you can set the value of **log@statistics.parameters** in the **Statistics log filename** field in the **Monitoring** tab of the **Gateway daemon settings** page.

# Chapter 45. Configuring bidirectional data support

Use the `ctg.bidi.target.layout` Java system property to enable right-to-left bidirectional (bidi) data support.

When right-to-left bidirectional (bidi) data support is enabled, bidi data in CHAR containers is converted from logical order to visual right-to-left order before the data is sent to CICS. The data is converted back from visual right-to-left order to logical order when the data is returned from CICS to CICS Transaction Gateway.

To enable right-to-left bidi data support use one of the following methods:

- For Java Client applications running in local mode, set the system property `ctg.bidi.target.layout` to `RTL`, as shown in the following example:

  ```
  java -Dctg.bidi.target.layout=RTL application
  ```

- For Client applications running in remote mode and connecting to CICS Transaction Gateway on UNIX or Linux, start CICS TG with the Java system property `ctg.bidi.target.layout` set to `RTL`. For example:

  ```
  ctgstart -j-Dctg.bidi.target.layout=RTL
  ```

- For Client applications running in remote mode and connecting to CICS Transaction Gateway on Windows, use the **ctgservice** command to specify the Java system property as follows:

  ```
  ctgservice -R -A-j-Dctg.bidi.target.layout=RTL
  ```

If you start your local mode Java Client application or CICS Transaction Gateway without the system property options described above, by default, bidi data support is off. However, you can explicitly turn off bidi support by specifying -
`Dctg.bidi.target.layout=OFF`

.

You cannot enable bidi data support through the Configuration Tool.

# Chapter 46. 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.

To create your own customized mapping file on Windows, copy <install_path>/samples/configuration/cicskeysamp.ini to <product_data_path>/cicskey.ini.

To create your own customized mapping file on UNIX and Linux, copy <install_path>/samples/configuration/cicskeysamp.ini to /var/cicscli/cicskey.ini.

The keyboard mapping file can be specified by:

- Using the -k option on the **cicsterm** command.
- Setting the *CICSKEY* environment variable. For example on Windows:

```
SET CICSKEY=C:/custom/mykeys.ini
```

  and on UNIX and Linux:

```
export CICSKEY=/var/cicscli/mykeys.ini
```

See your Windows operating system documentation for other ways of setting this environment variable.

On UNIX and Linux if you do not specify the -k option or the *CICSKEY* variable and do not create /var/cicscli/cicskey.ini, the keyboard mapping is defined by the default file <install_path>/bin/cicskey.ini.

On Windows if you do not specify the -k option or the *CICSKEY* variable and do not create <product_data_path>/cicskey.ini, the keyboard mapping is defined by the default file <install_path>/bin/cicskey.ini.

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 any one of the following on Windows or either `Ctrl` or `Shift` on UNIX and Linux:

```
Alt
Ctrl
Shift
```

The *Key* can be any one of the keys shown in Table 8, (but some combinations of modifier+key are not supported — see "Key combinations" on page 229):

*Table 8. 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`<br>`n o p q r s t u v w x y z` |
| Tab key | `Tab` |
| Movement keys | `newline backspace`<br>`insert home pageup`<br>`delete end pagedown`<br>`up-left-down-right` |

*Table 8. Keys that you can map (continued)*

| Group | Keys |
|---|---|
| Keypad keys | keypad/ keypad* keypad-<br>keypad7 keypad8 keypad9<br>keypad4 keypad5 keypad6 keypad+<br>keypad1 keypad2 keypad3<br>keypad0 keypad. keypadenter |

In addition, the following key mappings are allowed on Windows:

```
bind Clear         Pause
bind Reset         Scroll Lock
```

**Note:** For the Client daemon on UNIX and Linux, 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

This topic describes the valid combinations of modifier and keys that can be mapped.

**No modifier**
All keys available for mapping.

**Alt modifier (Windows)**
Only function keys, numeric keys, movement keys, and alphabetic keys can be mapped.

**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.

**Note:** All modifier and key combinations that are not pre-empted by Windows 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.

To create your own customized mapping file on Windows, copy `<install_path>/samples/configuration/cicskeysamp.ini` to `<product_data_path>/cicskey.ini`.

To create your own customized mapping file on UNIX and Linux, copy `<install_path>/samples/configuration/cicskeysamp.ini` to `/var/cicscli/cicskey.ini`.

The color mapping file can be identified by:

- Using the **-c** option on the **cicsterm** command. For more information see Chapter 71, “cicsterm command reference,” on page 379.
- Setting the *CICSCOL* environment variable: For example on Windows:

```
SET CICSCOL=C:/custom/mycols.ini
```

  and on UNIX and Linux:

```
export CICSCOL=/var/cicscli/mycols.ini
```

See your Windows operating system documentation for other ways of setting this environment variable.

On UNIX and Linux if you do not specify the **-c** option or the *CICSCOL* variable and do not create /var/cicscli/cicscol.ini, the keyboard mapping is defined by the default file <install_path>/bin/cicscol.ini.

On Windows if you do not specify the **-c** option or the *CICSCOL* variable and do not create <product_data_path>/cicscol.ini, the keyboard mapping is defined by the default file <install_path>/bin/cicscol.ini.

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.

## The color mapping file

The color mapping file is used to specify the screen colors for cicsterm.

The syntax of the color mapping file is shown below. Each binding must be on a separate line:

```
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.

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.

# Chapter 47. 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.

## Gateway trace file

The **tfile** parameter is the path name of the trace file where Gateway trace messages are written, if tracing is enabled.

**tfile=<pathname>**

**Description**

Set the value to the path name of the trace file. If you specify a file name without a path, the file is created in the <product_data_path> directory on Windows platforms and /var/cicscli directory on UNIX and Linux platforms. 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. Turning on the Gateway trace has a significant impact on performance.

You can use the **ctgservice -tfile=***pathname* on Windows or **ctgstart -tfile=***pathname* on UNIX and Linux command to override the value of **tfile**. For more information, see Command reference.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**

If this parameter is not specified, by default trace output is written to <product_data_path>/gateway.trc on Windows platforms and /var/cicscli/gateway.trc on UNIX and Linux platforms.

**Configuration Tool**

In the Configuration Tool, you can set the value of **tfile** in the **Gateway trace file** field in the **Trace setting** page. To find this page, click **Options** > **Trace Settings**.

## Gateway trace file wrap size (KB)

The **tfilesize** parameter specifies the maximum size, in kilobytes, of the Gateway trace file. When the file reaches this size, subsequent trace entries continue to be written from the beginning of the file.

**tfilesize=<number>**

**Description**

Set the value in the range 0 - 1,000,000 to specify the maximum trace file size. To disable wrapping, set the value to 0. If you set the value in the range 1 - 39, the CICS Transaction Gateway uses a value of 40 instead, to guarantee an adequate minimum trace size.

You can use the **ctgservice -tfilesize=***number* on Windows or **ctgstart -tfilesize=***number* on UNIX and Linux command to override the value of **tfilesize**. For more information, see Command reference.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**
If this parameter is not specified, the default value is 0.

**Configuration Tool**
In the Configuration Tool, you can set the value of **tfilesize** in the **Gateway trace file wrap size (KB)** field in the **Trace setting** page. To find this page, click **Options** > **Trace Settings**.

## Data byte offset in trace data

The **dumpoffset** parameter specifies the byte offset in data to start trace output.

**dumpoffset=<number>**

**Description**
Set the value to specify the number of bytes from which to start the trace output for a hex dump.

You can use the **ctgservice -dumpoffset=***number* command on Windows or the **ctgstart -dumpoffset=***number* command on UNIX and Linux to override the value of **dumpoffset**. For more information, see Chapter 69, "Command reference," on page 361.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**
If this parameter is not specified, the default value is 0.

**Configuration Tool**
This parameter cannot be set using the Configuration Tool.

## Maximum size of trace data blocks

The **truncationsize** parameter specifies the maximum size, in bytes, of the Gateway trace data blocks.

**truncationsize=<number>**

**Description**
Set the value to 0 or above to specify the size of the data blocks. If you specify 0, no data blocks are shown in the trace.

You can use the **ctgservice -truncationsize=***number* on Windows or the **ctgstart -truncationsize=***number* command on UNIX and Linux to override the value of **truncationsize**. For more information, see the Chapter 69, "Command reference," on page 361.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**
If this parameter is not specified, the default value is 80.

**Configuration Tool**
This parameter cannot be set using the Configuration Tool.

## Exception stack tracing

The **stack** parameter defines if exception stack tracing is enabled.

**stack=<on|off>**

**Description**
Set the value to **on** to enable exception stack tracing.

You can use the **ctgservice** command on Windows or the **ctgstart** command on UNIX and Linux with the **-stack** option to override the value of **stack**. For more information, see Command reference.

This parameter is in the "GATEWAY section of the configuration file" on page 240.

**Default value**
The default for this parameter is off.

**Configuration Tool**
This parameter cannot be set using the Configuration Tool.

## Client trace file

The **TRACEFILE** parameter specifies the file where Client trace messages are written, if tracing is enabled.

**TRACEFILE=<filepath>**

**Description**
Specify the file name or fully qualified file name of the trace file. If you do not include the path information, the file is created in the directory `<product_data_path>` on Windows platforms and in `/var/cicscli` on UNIX and Linux platforms. 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 formatting the trace file, see "Formatting the binary trace file" on page 502.

This parameter is in the "CLIENT section of the configuration file" on page 244.

**Default value**
If this parameter is not specified, by default trace output is written to `<product_data_path>\cicscli.bin` on Windows platforms and to `/var/cicscli/cicscli.bin` on UNIX and Linux platforms.

**Configuration Tool**
In the Configuration Tool, you can set the value of **TRACEFILE** in the **Client trace file** field on the **Trace settings** page. To find this page, click **Options** > **Trace Settings**.

## Client trace file wrap size

The **maxwrapsize** parameter determines the amount of storage that is reserved for trace data.

**maxwrapsize=<number>**

**Description**

Set the value in the range 0 - 2,000,000 to define the amount of storage reserved in kilobytes. If you specify 0, wrapping trace is disabled. If you specify a value in the range 1 - 99, then 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 trace is written directly to disk 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, overwriting the existing records.

This parameter is in the "CLIENT section of the configuration file" on page 244.

**Default value**

If this parameter is not specified, the default value is 0.

**Configuration Tool**

In the Configuration Tool, you can set the value of **maxwrapsize** in the **Client trace file wrap size (KB)** field on the **Trace settings** page. To find this page, click **Options** > **Trace Settings**. Alternatively, you can specify that all the components are traced by selecting the **Enable Gateway daemon trace and select all component trace modules** check box on the same page.

## Client trace components

The **TRACE** parameter specifies the Client components to trace when tracing is enabled.

**TRACE=<list>**

**Description**

Specify a comma-separated of all the Client components to trace. The possible 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. |

| Configuration tool fields | ctg.ini file trace parameters | Description |
|---|---|---|
| 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. |
| Workload manager | LMG | Workload Manager |

You can also use the `-m` parameter of the `cicscli` command to specify to specify Client trace components. This overrides any settings in the configuration file. For more information about the `cicscli` command, see Chapter 66, "Client daemon administration," on page 349.

This parameter is in the "CLIENT section of the configuration file" on page 244.

**Default value**

If you enable tracing without specifying the components in either the `cicscli` command or in the configuration file, the following set of components are traced:

- Protocol drivers
- The Client daemon
- Client API level 1
- Transport layer level 1

**Configuration Tool**

In the Configuration Tool, you can specify the list of Client components using the check boxes in the **Client component trace** section of the **Trace settings** page. To find this page, click **Options** > **Trace Settings**. Alternatively, you can specify that all the Client components are traced and Gateway daemon tracing is enabled by selecting the **Enable Gateway daemon trace and select all component trace modules** check box on the same page.

## 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 Chapter 65, "Gateway daemon administration," on page 343 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 `/var/cicscli` on UNIX and Linux or `<product_data_path>` on Windows.

You cannot enable JNI trace through the Configuration Tool.

# Chapter 48. 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`. You can specify a different location and optionally change the name of the configuration file by setting the *CICSCLI* environment variable. For more information see "Specifying the configuration file" on page 95.

To create a new configuration, either use the Configuration Tool, or edit a copy of the sample configuration file supplied in `<install_path>/samples/configuration/ctgsamp.ini`. Save the new configuration file in one of these locations:

- The `<product_data_path>` directory on Windows, using the default file name.
- The `/var/cicscli` directory on UNIX and Linux, using the default file name
- A directory and filename specified in the *CICSCLI* environment variable

For more information about the Configuration Tool see Chapter 31, "Creating a configuration file," on page 93.

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 or 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, with only 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

This table provides the names and descriptions for all parameters that can be set in the PRODUCT section of the configuration file.

The PRODUCT section of the configuration file defines product wide settings. There must be no more than one PRODUCT section in the configuration file.

*Table 9. SECTION PRODUCT*

| Entry in the configuration file | Description |
|---|---|
| applid | "Gateway APPLID" on page 107 |
| applidqualifier | "Gateway APPLID qualifier" on page 108 |

*Table 9. SECTION PRODUCT (continued)*

| Entry in the configuration file | Description |
|---|---|
| defaultserver | "Default server" on page 213 |
| keyring | "Key ring file" on page 206 |
| keyringpw | "Key ring password" on page 206 |
| keyringpwscrambled | "Key ring password encryption" on page 206 |

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

The GATEWAY section of the configuration file defines the Gateway daemon settings.

There must be no more than one GATEWAY section in the configuration file.

*Table 10. SECTION GATEWAY*

| Entry in the configuration file | Description |
|---|---|
| adminport | "Port for local administration" on page 161. |
| cicsintercept | Chapter 35, "Configuring for application testing," on page 105 |
| cicslogging | "Log CICS messages" on page 167. |
| closetimeout | "Timeout for in-progress requests to complete" on page 160. |
| connectionlogging | "Log Client connections and disconnections" on page 167. |
| DNSNames | "Display TCP/IP host names" on page 168. |
| dumpoffset | "Data byte offset in trace data" on page 234. |
| initconnect | "Initial number of connection manager threads" on page 157. |
| initworker | "Initial number of worker threads" on page 158. |
| log@error.dest | Log destinations. |
| log@error.parameters | Log file names. |
| log@info.dest | "Informational message logging" on page 164. |
| log@info.parameters | "Informational message logging" on page 164. |
| log@statistics.dest | "Statistics log destination" on page 223. |
| log@statistics.parameters | "Statistics log file name" on page 224. |
| maxconnect | "Maximum number of connection manager threads" on page 157. |
| maxhttpconnect | "Maximum number of HTTP connections" on page 159 |
| maxworker | "Maximum number of worker threads" on page 158. |
| noinput (UNIX and Linux) | "Enable reading input from console on UNIX and Linux" on page 160. |
| quiet (UNIX and Linux) | . |
| requestexits | "Configuring request monitoring exits for the Gateway daemon" on page 219. |
| stack | "Exception stack tracing" on page 235. |
| stateod | "Statistics end of day time" on page 222. |
| statint | "Statistics interval" on page 222. |
| statsrecording | "Enable statistic recording to an XML file" on page 223. |

*Table 10. SECTION GATEWAY (continued)*

| Entry in the configuration file | Description |
|---|---|
| trace | Equivalent to Gateway under "Client trace components" on page 236. Set this to **on** to enable Gateway trace, otherwise omit. |
| tfile | "Gateway trace file" on page 233. |
| tfilesize | "Gateway trace file wrap size (KB)" on page 233. |
| trace | "Gateway daemon tracing" on page 499. |
| truncationsize | "Maximum size of trace data blocks" on page 234. |
| workertimeout | "Worker thread availability timeout" on page 159. |

## 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 11. TCP protocol*

| Entry in the ctg.ini file | Description |
|---|---|
| bind | "Bind address" on page 169 |
| connecttimeout | "Connection timeout" on page 169 |
| dropworking | "Drop working connections" on page 171 |
| idletimeout | "Idle timeout" on page 170 |
| pingfrequency | "Ping frequency interval" on page 170 |
| port | "Port" on page 169 |
| requiresecurity | "Require Java Clients to use security classes" on page 172 |
| solinger | "SO_LINGER setting" on page 171 |

## 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
```

Followed by:

```
protocol@ssl.parameters=clientauth=<on>;connecttimeout=<number>;\
dropworking;idletimeout=<number>;\
pingfrequency=<number>;port=<number>;requiresecurity;solinger=<number>;\
ciphersuites=<name>;
```

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 12. SSL protocol*

| Entry in the configuration file | Description |
|---|---|
| bind | "Bind address" on page 172. |
| ciphersuites | "Use only these ciphers" on page 176. |
| clientauth | "Use client authentication" on page 176. |
| connecttimeout | "Connection timeout" on page 169. |
| dropworking | "Drop working connections" on page 171. |
| idletimeout | "Idle timeout" on page 170. |
| pingfrequency | "Ping frequency interval" on page 170. |
| port | "Port" on page 173. |
| requiresecurity | "Require Java Clients to use security classes" on page 172. |
| solinger | "SO_LINGER setting" on page 171. |

## HTTP protocol parameters

To enable the HTTP protocol, add a HTTP subsection to the `ctg.ini` configuration file.

In the GATEWAY section of the configuration file, add the following subsection:

```
SUBSECTION HTTP
 port=<number>
 bind=<name>
ENDSUBSECTION
```

Entries correspond to fields in the HTTP settings panel:

*Table 13. HTTP protocol*

| Entry in the ctg.ini file | Description |
|---|---|
| bind | "Bind address" on page 193 |
| port | "Port" on page 193 |

## HTTPS protocol parameters

To enable the HTTPS protocol, add a HTTPS subsection to the `ctg.ini` configuration file.

In the GATEWAY section of the configuration file, add the following subsection:

```
SUBSECTION HTTPS
 port=<number>
 bind=<name>
 ciphersuites=<name>
 clientauth=<on>
ENDSUBSECTION
```

Entries correspond to fields in the HTTPS settings panel:

*Table 14. HTTPS protocol*

| Entry in the configuration file | Description |
|---|---|
| bind | "Bind address" on page 194 |
| ciphersuites | "Use only these ciphers" on page 195 |
| clientauth | "Use client authentication" on page 195 |
| port | "Port" on page 194 |

## 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 15. Statistics API protocol parameters*

| Entry in the ctg.ini file | Description |
|---|---|
| bind | "Bind address" on page 220 |
| connecttimeout | "Connection timeout" on page 169 |
| port | "Port" on page 221 |
| maxconn | "Maximum number of connections" on page 222 |

## CLIENT section of the configuration file

The CLIENT section of the configuration file defines the Client daemon settings.

There must be no more than one CLIENT section in the configuration file.

*Table 16. SECTION CLIENT*

| Entry in the configuration file | Description |
|---|---|
| SECTION CLIENT | Setting SECTION CLIENT = *Application ID* is deprecated and supported only for compatibility with earlier versions. Application ID has been replaced by the APPLID parameter, which now takes precedence. See Chapter 36, "Configuring identification using APPLID," on page 107 for more information.<br>**Note:** When creating a new configuration file, set SECTION CLIENT = *. |
| ccsid | CCSID |
| enablepopups (Windows) | "Enable pop-up windows" on page 183 |
| loadmanager (Windows) | Enable the Workload Manager |
| logfile | "Error and warning log file" on page 185 |
| logfileinfo | "Information log file" on page 186 |
| maxbuffersize | "Maximum buffer size" on page 179 |
| maxrequests | "Maximum requests" on page 180 |
| maxservers | "Maximum servers" on page 180 |
| maxwrapsize | "Client trace file wrap size" on page 235 |
| printcommand | "Print command" on page 181 |
| printfile | "Print file" on page 181 |
| srvretryinterval | "Server retry interval" on page 182 |
| terminalexit | "Terminal exit" on page 179 |
| terminstlogging | "Log terminal installations and deletions" on page 186 |
| trace | "Client trace components" on page 236 |
| tracefile | "Client trace file" on page 235 |
| useoemcp (Windows) | "Use OEM code page" on page 184 |

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

An IPICSERVER section in the configuration file defines a CICS server to which the Gateway daemon can connect over IPIC.

An IPICSERVER section definition is required for each CICS server to be connected using the IPIC protocol.

*Table 17. SECTION IPICSERVER*

| Entry in the configuration file | Description |
|---|---|
| SECTION IPICSERVER | "Server name" on page 116 |
| cicsapplid | "Target CICS APPLID" on page 119 |

*Table 17. SECTION IPICSERVER (continued)*

| Entry in the configuration file | Description |
|---|---|
| cicsapplidqualifier | "Target CICS APPLID qualifier" on page 119 |
| connecttimeout | "Connection timeout" on page 120 |
| description | "Description" on page 117 |
| ecitimeout | "ECI timeout" on page 122 |
| hostname | "Host name or IP address" on page 118 |
| heartbeatinterval | "IPIC Heartbeat interval" on page 122 |
| port | "Port" on page 118 |
| sendsessions | "IPIC send sessions" on page 118 |
| srvidletimeout | "Server idle timeout" on page 121 |
| srvretryinterval | "Server retry interval" on page 120 |
| tcpkeepalive | "Send TCP/IP KeepAlive packets" on page 121 |
| ssl | "Use SSL" on page 123 |
| ciphersuites | "Use only these ciphers" on page 123 |

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

A SERVER section in the configuration file defines a server to which the Client daemon can connect.

A SERVER section definition is required for each CICS server to be connected using the TCP/IP or SNA protocol.

*Table 18. SECTION SERVER*

| Entry in the configuration file | Description |
|---|---|
| SECTION SERVER | "Server name" on page 126. |
| description | "Description" on page 117. |
| initialtransid | "Initial transaction" on page 127. |
| modelterm | "Model terminal definition" on page 127. |
| protocol | "CICS server connection protocols" on page 128. |
| region (Windows) | "Configuring a server group" on page 214. |
| uppercasesecurity | "Use uppercase security" on page 130. |

Other entries in this section depend upon the protocol selected.

*Table 19. SECTION SERVER: additional entries for the TCP protocol*

| Entry in the configuration file | Description |
|---|---|
| connecttimeout | "Connection timeout" on page 129. |
| netname | "Host name or IP address" on page 128. |
| port | "Port" on page 128. |
| srvidletimeout | "Server idle timeout" on page 129. |

*Table 19. SECTION SERVER: additional entries for the TCP protocol (continued)*

| Entry in the configuration file | Description |
|---|---|
| tcpkeepalive | "Send TCP/IP KeepAlive packets" on page 130. |

*Table 20. SECTION SERVER: additional entries for the SNA protocol*

| Entry in the configuration file | Description |
|---|---|
| locallualias (Windows) | "Use local LU alias name" on page 152. |
| localluname | "Local LU name" on page 152. |
| modename | "Mode name" on page 140. |
| netname | "Partner LU name" on page 153. |
| partnerlualias | "Use Partner LU alias name" on page 153. |
| srvidletimeout | "Server idle timeout" on page 129. |

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 sections of the configuration file determine which communications protocol DLL is used for communicating with a CICS server connected using the TCP/IP or SNA protocol.

At least one DRIVER section must exist in the configuration file for each CICS server connection protocol used with your CICS servers. The DRIVER sections of the configuration file must include a **SECTION DRIVER** and **DRIVERNAME** parameter. The syntax for the DRIVER sections must match the following example:

```
SECTION DRIVER=<protocol>
 DRIVERNAME=<DLL>
ENDSECTION
```

Use of the backslash (\) character to split lines in the DRIVER sections of the configuration file is not supported.

## SECTION DRIVER

The **SECTION DRIVER** parameter defines the CICS server connection protocol used to connect to your CICS servers.

**SECTION DRIVER=<protocol>**

**Description**
Set the value to either SNA or TCPIP. The value specified must match the value specified in the **protocol** parameter.

This parameter is in the "DRIVER section of the configuration file."

**Default value**
There is no default value for this parameter.

**Configuration Tool**
This parameter cannot be set using the Configuration Tool. The Configuration Tool automatically selects the correct protocol.

## Driver DLL

The **DRIVERNAME** parameter defines the Windows, UNIX, or Linux DLL for the specified CICS server connection protocol.

**DRIVERNAME=<DLL>**

**Description**

Set the value to the appropriate DLL for the protocol specified in the **SECTION DRIVER** parameter. The following list details the appropriate DDL for each protocol:

- For SNA protocol: Specify CCLWNTSN for Windows or CCLIBMSN for UNIX and Linux.
- For TCPIP protocol: Specify CCLWNTIP for Windows or CCLIBMIP for UNIX and Linux.

This parameter is in the "DRIVER section of the configuration file" on page 246.

**Default value**

There is no default value for this parameter.

**Configuration Tool**

This parameter cannot be set using the Configuration Tool. The Configuration Tool automatically selects the correct DLL value.

# LOADMANAGER section of the configuration file

The LOADMANAGER section of the configuration file defines Windows Workload Manager settings.

The configuration of the Workload Manager requires a LOADMANAGER section to be defined together with region parameters in the SERVER sections. See "Enable the Workload Manager" on page 214 for more information.

*Table 21. SECTION LOADMANAGER*

| Entry in the configuration file | Description |
|---|---|
| bias | "Configuring workload biasing" on page 215 |
| nobalance | "Disabling workload balancing for a program" on page 217 |
| notmanaged | Manage workload for this server group |
| timeout | "Server group timeout" on page 216 |
| type | "Round robin or biasing" on page 215 |

Use of the backslash (\) character to split lines in the LOADMANAGER section of the configuration file is not supported.

# WEBSERVICE section of the configuration file

A WEBSERVICE section in the configuration file defines a web service. A WEBSERVICE section is required for each web service exposed by CICS TG.

A WEBSERVICE section is required for each JSON web service that the Gateway daemon makes available. Each section requires a unique name. If you have two WEBSERVICE sections with the same name, the Gateway fails to start and displays error message CTG8466E.

The name of the WEBSERVICE must be made up of characters acceptable in a URI[1] or the escape coded versions of that character.

*Table 22. SECTION WEBSERVICE*

| **Entry in the configuration file** | **Description** |
|---|---|
| SECTION WEBSERVICE | “Web service name” on page 189 |
| bindfile | “Bind file location” on page 191 |
| defaultmirror | “Use default mirror transaction” on page 192 |
| server | “Server name” on page 191 |
| timeout | “Timeout” on page 192 |
| transactionid | “Transaction identifier” on page 191 |
| uri | “Uri” on page 190 |

# Chapter 49. Environment variables reference

The table lists the environment variables for controlling how CICS Transaction Gateway functions.

Environment variables and the configuration file control how the CICS Transaction Gateway functions. When an environment variable is intended for use with the Gateway daemon or Client daemon the variable must be defined as a system environment variable and the system must be restarted when the variable is first set or changed.

*Table 23. Environment variables*

| Environment variable | Description |
|---|---|
| CICSCLI | See "Specifying the configuration file" on page 95 |
| CLASSPATH | On UNIX and Linux set the *CLASSPATH* variable for the resource adapter Java classes. You must set the Java *CLASSPATH* environment variable so that the resource adapter Java classes can be used when an ECI request is sent to the CICS server.<br><br>On Windows, the *CLASSPATH* variable is updated by the installation process defining product jar files used by the CICS Transaction Gateway. |
| CTGDCONF (UNIX and Linux) | See "Running the Gateway daemon as a background process" on page 333. |
| CTG_DATA_PATH (Windows) | Created by the installation process with the value of <product_data_path>. See Chapter 20, "File path terminology," on page 57 for more information. |
| CTG_JAVA | See "Specifying the JVM" on page 95. |
| CTG_JNI_TRACE | Specify the name of the JNI trace file. See "JNI tracing" on page 506. |
| CTG_JNI_TRACE_ON | Switch JNI tracing on. See "JNI tracing" on page 506. |
| PATH (Windows) | Updated by the installation process to append `<install_path>\bin`. |
| LD_PRELOAD (UNIX and Linux) | Configuring IBM Communications Server for Linux. |

# Chapter 50. 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 adapter can be used with your JEE 6, 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 the ECI resource adapter can be used with a particular application server.

The IVT can be used to verify the use of the ECI resource adapter with a JEE 6, or later, 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 adapter 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 JEE 6 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: Developing Applications* to run the EciB1 sample program. The sample program can be found in <install_path>/samples.

## Deploying and configuring the JCA IVT

To deploy and configure the JCA IVT, you install the resource adapter archive file (.rar), define a connection factory, and set the connection factory properties.

To do this, complete the following tasks:

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 the port number to the port number of the relevant Gateway daemon protocol handler.

   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`. 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. If you are using WebSphere Application Server 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: Developing ApplicationsCICS Transaction Gateway for IBM z/OS: Developing Applications* for details, including compilation instructions and information on compiler considerations.

# Part 7. 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.

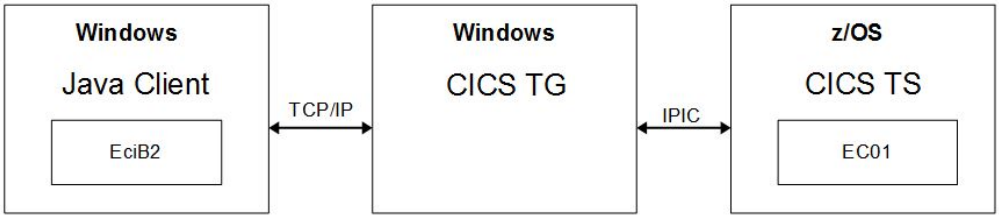
The prerequisites topic in each scenario describes the minimum product requirements for that scenario, although in some cases, a later release may be used in the actual example.

# Chapter 51. Configuring a secure autoinstalled IPIC connection (SC01)

You can configure secure autoinstalled IPIC connections using a template.

You use an IPCONN template to change the default connection settings, so that IPIC connections are autoinstalled with link security and user security. To configure secure autoinstalled IPIC connections, you must modify the CICS TS sample user-replaceable module (URM) to point to an IPCONN template.

The following figure shows the topology used in this scenario:




**Note:** This scenario uses CICS Transaction Gateway connecting to CICS Transaction Server V5.3 over IPIC in remote mode. You can run this scenario with other versions, but the illustrations and samples might vary. For the minimum requirements, see "Prerequisites" on page 258.

This scenario uses the default name `ctg.ini` for the configuration file.

*Table 24. 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 |

*Table 24. Values used in this scenario (continued)*

| Component | Parameter | Where set | Example value | Matching values |
|---|---|---|---|---|
| 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 | |

The sample configuration file `ctg.ini` for this scenario is provided in the directory `<install path>/samples/scenarios/sc01`.

## Prerequisites

You must satisfy these system requirements.

Here are the system requirements for CICS TS for IBM 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 TCPIP 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.
- You must set the SEC system initialization parameter to YES to enable security.
- You must have valid IBM 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 IBM 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      ==> 00010             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( 00010 ) Max( 000000 ) Urm( DFHISCIP )
     ```

**Note:** You can configure CICS resources using the CICS Explorer , see the CICS Explorer information in the CICS TS documentation 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, start the CICS Transaction Gateway and 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(00000006) App(00000006)              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)
```

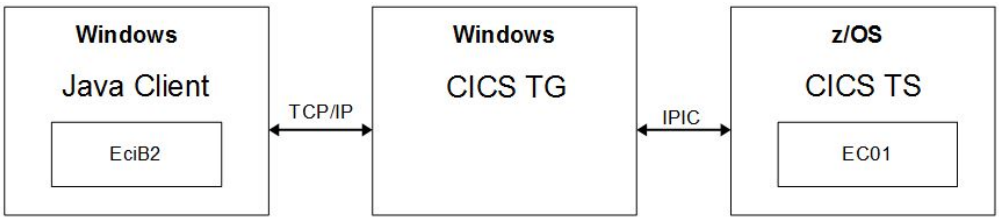
Note that the IPCONN template must be INS REL for it to be used by an incoming request. The autoinstalled IPCONN, for example, APPL, is INS ACQ.

# Chapter 52. Configuring a secure predefined IPIC connection (SC02)

You can configure a secure predefined IPCONN for your IPIC connection between CICS TG and CICS TS.

A predefined IPCONN provides a more secure environment and can prevent unwanted IPCONN autoinstall requests. To configure a secure predefined IPCONN for your IPIC connection between CICS TG and CICS TS, follow the instructions in this scenario.

The following figure shows the topology used in this scenario:




**Note:** This scenario uses CICS Transaction Gateway connecting to CICS Transaction Server V5.3 over IPIC in remote mode. You can run this scenario with other versions, but the illustrations and samples might vary. For the minimum requirements, see "Prerequisites" on page 266.

This scenario uses the default name `ctg.ini` for the configuration file.

*Table 25. 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 |

*Table 25. Values used in this scenario (continued)*

| Component | Parameter | Where set | Example value | Matching values |
|---|---|---|---|---|
| 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 |
| 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 | |

The sample configuration file `ctg.ini` for this scenario is provided in the directory `<install path>/samples/scenarios/sc02`.

## Prerequisites

You must satisfy these system requirements.

Here are the system requirements for CICS TS for IBM 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.
- You must set the SEC system initialization parameter to YES to enable security.
- You must have valid IBM 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      ==> 00010     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( 00010 ) 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, start the CICS Transaction Gateway and 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

   Call Programs

   About to call : EC01
     COMMAREA    :
     extend_Mode : 0
     LUW_Token   : 0
     Commarea    : 24/02/16 18:51:34
   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( 00001 ) 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=IY2GTGXX
             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.
```

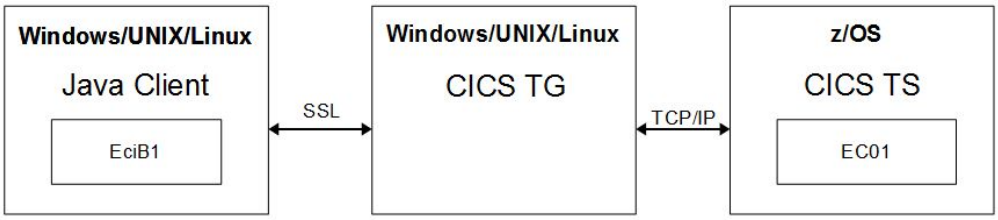
# Chapter 53. Configuring SSL security between a Java Client and the Gateway daemon (SC06)

This scenario shows how to configure SSL on the connection between a Java client running on Windows, UNIX, or Linux and CICS Transaction Gateway for Multiplatforms. The connection between CICS Transaction Gateway and CICS Transaction Server for IBM 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, and send an ECI request to the CICS server to check that the SSL connection works.

In this scenario, when the Java client attempts to connect to the Gateway daemon's SSL protocol handler, an SSL handshake between the Java client and the Gateway daemon is performed to authenticate the server and to establish the cryptographic keys which are used to protect the data to be transmitted. The scenario includes an optional step where the Gateway daemon requests the Java client to authenticate itself by providing its public key and digital certificate. This is known as client authentication.

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 | In the protocol@ssl.parameters parameters in the SECTION GATEWAY in ctg.ini | on |
| CICS TG | keyring | SECTION PRODUCT in ctg.ini | MyServer.jks |
| CICS TG | keyringpw | SECTION PRODUCT in ctg.ini | MyPassword |
| CICS TG | port | In the protocol@ssl.parameters parameters in the SECTION GATEWAY in ctg.ini | 8573 |
| CICS TG | server | SECTION SERVER in ctg.ini | CICSA |
| CICS TG | protocol | SECTION SERVER in ctg.ini | TCPIP |

| Component | Parameter | Where set | Example value |
|---|---|---|---|
| CICS TG | netname | SECTION SERVER in ctg.ini | cicssrv1.company.com |
| CICS TG | port | SECTION SERVER in ctg.ini | 7760 |

The sample configuration file `ctg.ini` for this scenario is provided in the directory `<install path>/samples/scenarios/sc06`.

## Prerequisites

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 IBM z/OS:

- TCP/IP services must be active in the CICS server.
  - To activate these services, set the TCPIP 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.
- 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 124.

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 provided as part of the Java Runtime Environment.

For information about the benefits of using SSL see "Why use SSL?" on page 397.

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, click **Add**.
2. Select the certificate name **exampleservercert.arm**.
3. Click **OK**.
4. In the Certificate file name field type a unique, recognizable name, for example, `my self-signed server authority`.
5. 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 into the server keyring.

iKeyman is provided as part of the Java Runtime Environment.

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 key ring `MyServer.jks`. This key ring 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 key ring requires the client certificate as a signer certificate. In this scenario, the client certificate is `exampleclientcert` and in the server key ring, `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. Update the PRODUCT section definition:
   a. Set **keyring** to `MyServer.jks`.

      This is the name of the key ring to be used by this SSL protocol handler. For more information, see "Key ring file" on page 206.

   b. Set **keyringpw** to `MyPassword`.

      This is the password that you used for the server key ring. For more information, see "Key ring password" on page 206.

   When you have made these updates, the PRODUCT section of your configuration file should contain the following definitions:

   ```
   SECTION PRODUCT
       KeyRing=MyServer.jks
       KeyRingPw=MyPassword
   ENDSECTION
   ```

3. Update the SSL protocol handler parameters:
   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 **port** to `8573`.

      This parameter identifies the TCP/IP port on which the protocol handler listens for incoming client requests.

   When you have made these updates, the SSL protocol handler parameters definition in your configuration file should contain the following definition:
4. 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:

```
CTG8489I The following cipher suites are provided by JSSE:
TLS_EMPTY_RENEGOTIATION_INFO_SCSV
SSL_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
SSL_ECDHE_RSA_WITH_AES_128_CBC_SHA256
SSL_RSA_WITH_AES_128_CBC_SHA256
SSL_ECDH_ECDSA_WITH_AES_128_CBC_SHA256
SSL_ECDH_RSA_WITH_AES_128_CBC_SHA256
SSL_DHE_RSA_WITH_AES_128_CBC_SHA256
SSL_DHE_DSS_WITH_AES_128_CBC_SHA256
SSL_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
SSL_ECDHE_RSA_WITH_AES_128_CBC_SHA
SSL_RSA_WITH_AES_128_CBC_SHA
SSL_ECDH_ECDSA_WITH_AES_128_CBC_SHA
SSL_ECDH_RSA_WITH_AES_128_CBC_SHA
SSL_DHE_RSA_WITH_AES_128_CBC_SHA
SSL_DHE_DSS_WITH_AES_128_CBC_SHA
SSL_ECDHE_ECDSA_WITH_RC4_128_SHA
SSL_ECDHE_RSA_WITH_RC4_128_SHA
SSL_RSA_WITH_RC4_128_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 **`set CLASSPATH`** or **`export CLASSPATH`** command, for example:
   - On UNIX and Linux:

     ```
     export CLASSPATH=<install_path>/classes/ctgclient.jar:
     <install_path>/classes/ctgsamples.jar:$CLASSPATH
     ```

   - On Windows:

     ```
     set 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:

On UNIX and Linux:

```
/opt/ibm/cicstg/samples/java/com/ibm/ctg/samples/eci
```

On Windows:

```
C:\Program Files\IBM\CICS Transaction Gateway\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
```

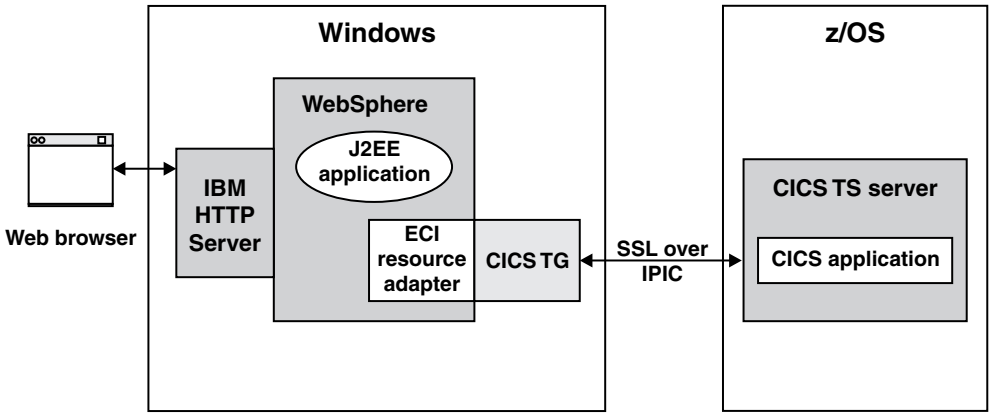
You have now completed the scenario.

# Chapter 54. Configuring SSL between CICS TG and CICS (SC07)

This scenario shows you how to configure SSL security on an IPIC connection.

**Note:** This scenario uses CICS Transaction Gateway running in local mode on IBM WebSphere Application Server V8.0 and CICS Transaction Server V5.3. You can run this scenario with other versions, but the illustrations and samples might vary. For the minimum requirements, see "Prerequisites" on page 282.

The following figure shows the topology used in this scenario:




Follow the 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 | IBM WebSphere Application Server | SSLAH |

## Prerequisites

Before you can complete this scenario, you must ensure that the system requirements for CICS Transaction Server, CICS Transaction Gateway, and IBM 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 TCPIP 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.

IBM WebSphere Application Server:

- IBM 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 install the sample CICS COBOL programs EC01, EC03 on the CICS server.

For information about the samples see COBOL, C and Map samples.

## Configuring SSL server authentication on the CICS server

To complete this task you use IBM 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 (IBM 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. These sample commands generate RACF certificates which use an SHA-256 hashing algorithm for signing. For information on other options see the topic RACDCERT GENCERT (Generate certificate) in the *IBM z/OS Security Server RACF Command Language Reference*.

1. Create a CA certificate that is self-signed on the server (in IBM RACF). Enter the following command on one line:

   ```
   RACDCERT CERTAUTH GENCERT SUBJECTSDN(OU('CTG TEST') O('IBM') T('CTG CA CERT') C('GB'))
   SIZE(2048) KEYUSAGE(CERTSIGN) WITHLABEL('CTG CA CERT')
   ```

2. Refresh the IBM 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. Enter the following command on one line::

```
RACDCERT ID(CTGUSER) GENCERT SUBJECTSDN(OU('CTG TEST') O('IBM') T('CTG PERSONAL CERT') C('GB'
SIZE(2048) 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(CE
```

b. Add the personal certificate to the keyring:

```
RACDCERT ID(CTGUSER) CONNECT(LABEL('CTG PERSONAL CERT') RING(CTGSERVERKEYRING) DEFAULT USA
```

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(C
```

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 provided as part of the Java Runtime Environment.

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 282, 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 provided as part of the Java Runtime Environment.

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 284.
   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        : 00010              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        : 050435363738392F303132330A1613100D0915120F0C03060201
 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        : 050435363738392F303132330A1613100D0915120F0C03060201
 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 IBM WebSphere Application Server" on page 289.

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 IBM 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 IBM WebSphere Application Server

To complete this task you use the IBM 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 IBM WebSphere Administrative Console, click **Resources** > **Resource Adapters**, click **Install RAR**.
   b. Click Browse for the Local file system. Select the CICS ECI resource adapter cicseci.rar. Take note of the scope; this choice limits the scope of later connection factory definitions. In this scenario, the scope is Node.
   c. Click **Next**. The resource adapter General Properties dialog contains a predefined name and description for the CICS ECI resource adapter. Leave the class path as it is currently set, and leave the native library path blank.
   d. Click **OK**.
   e. Save the changes to the master configuration.
2. Create and configure a J2C connection factory:
   a. In the IBM WebSphere Administrative Console, click **Resources** > **Resource Adapters**. Click on the ECIResourceAdapter.
   b. Under **Additional Properties** click on **J2C connection factories**.
   c. Click **New**.
   d. 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.
   e. Click **OK**.
   f. Click the new J2C connection factory CF-20.
   g. Click **Additional Properties** > **Custom Properties**.
   h. In the **Value** column of the Custom properties table, enter the values shown in the following table:

| Name | Value | Description | Required |
|---|---|---|---|
| You can administer the following resources: | | | |
| tPNName | CPMI | The transaction identifier of the CICS mirror transaction | false |
| applid | SSLAH | The APPLID for application using this connection | false |
| applidQualifier | | The APPLID qualifier for applications using this connection | false |
| cipherSuites | | The cipher suites available for an SSL connection | false |
| clientSecurity | | The class name of the client security exit for this connection | false |
| connectionURL | local: | The URL of the CICS Transaction Gateway for this connection | false |
| ipicHeartbeatInterval | 30 | For local mode, the heartbeat interval when using an IPIC connection | false |
| ipicSendSessions | 100 | For local mode, the number of simultaneous transactions or CICS tasks that are allowed over the connection when using an IPIC connection | false |
| keyRingClass | clientkeyring.jks | The location of the keystore containing the certificates required for an SSL connection | false |

| | | | |
|---|---|---|---|
| keyRingPassword | ****** | The password required to access the keystore for an SSL connection | false |
| password | | The default password or password phrase that requests through this connection use | false |
| portNumber | 2006 | The port number of the CICS Transaction Gateway for this connection | false |
| requestExits | | The class name of the request exits called during the execution of interactions | false |
| serverName | ssl://cicsserver:port | The name of the target CICS server for this connection | false |
| serverSecurity | | The class name of the server security exit for this connection requires the Gateway daemon to use | false |
| socketConnectTimeout | 0 | The number of milliseconds to wait while connecting to a Gateway daemon | false |
| traceLevel | 1 | The level of CICS Transaction Gateway diagnostic trace detail | false |
| tranName | | The transaction identifier placed in EIBTRNID by CICS for the mirror transaction | false |
| userName | CTGUSER | The default user name that requests through this connection use | false |
| xaSupport | off | This connection uses XA transactions | false |
| Total 20 | | | |

You do not have to supply a CICS password in the password field because the IPIC connection is qualified with AttachSec=Identify.

i. 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 reference to your connection factory JNDI name eis/CF-20:

| Select | Module | EJB | URI | Resource Reference | Target Resource JNDI Name | Login configuration |
|---|---|---|---|---|---|---|
| ☐ | ECIIVTEJB | ECIIVT | ECIIVTEJB.jar,META-INF/ejb-jar.xml | ECI | eis/CF-20 Browse... | Resource authorization: Per application |

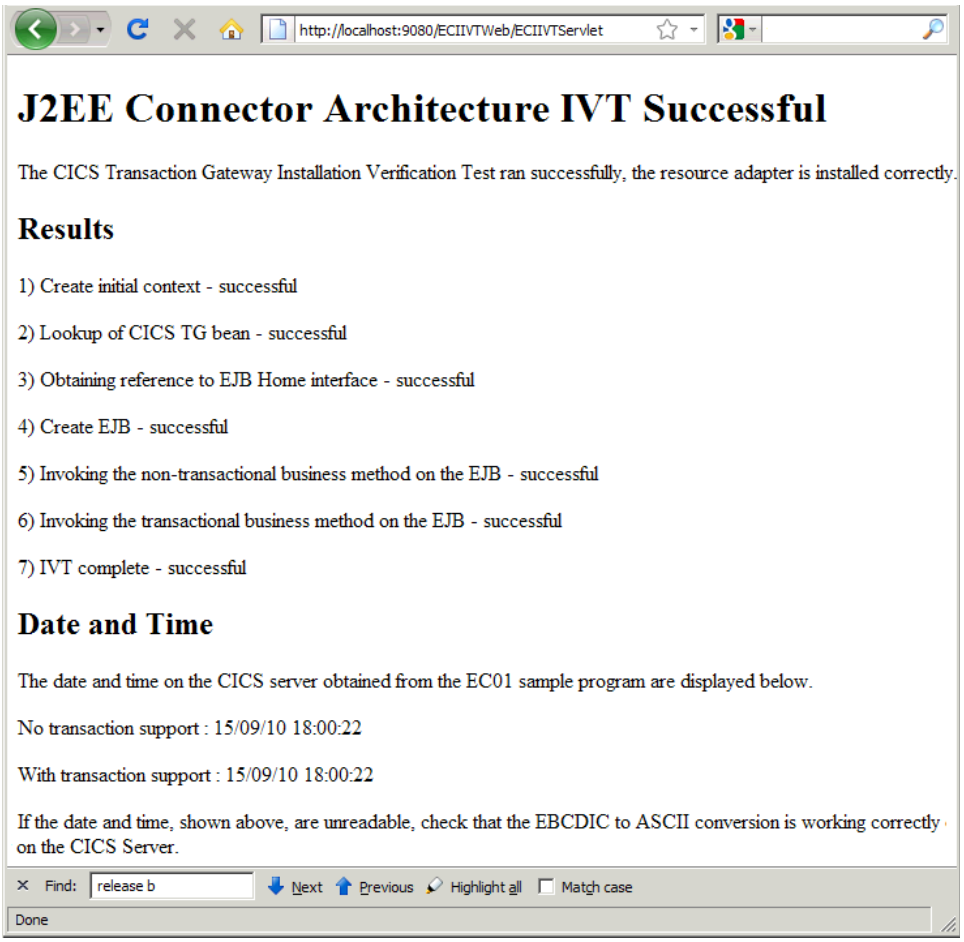
c. Save your configuration.
d. Restart IBM WebSphere Application Server if necessary (this depends on the version of WebSphere Application Server you are using).

You have now configured IBM 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 JEE 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 IBM WebSphere Application Server:

a. From the IBM 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 IBM WebSphere Application Server if necessary.

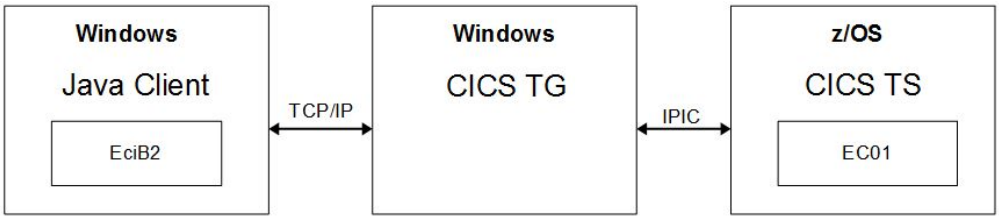
f. Look for the CICS Transaction Gateway trace in the `systemerr.log` file.

You have now completed the scenario.

# Chapter 55. Configuring an autoinstalled IPIC connection (SC08)

You can configure autoinstalled IPIC connections by using the default connection settings for IPIC autoinstalled connections. To implement IPIC connections that are autoinstalled without security, follow the step-by-step instructions in this scenario.

The following figure shows the topology used in this scenario:




**Note:** This scenario uses CICS TG connecting to CICS TS V5.3 over IPIC in remote mode. You can run this scenario with other versions, but the illustrations and samples might vary. For the minimum requirements, see "Prerequisites" on page 296.

This scenario uses the default name `ctg.ini` for the configuration file.

*Table 26. 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 | 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 |

The sample configuration file `ctg.ini` for this scenario is provided in the directory `<install path>/samples/scenarios/sc08`.

## Prerequisites

You must satisfy these system requirements.

Here are the system requirements for CICS TS for IBM 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 TCPIP 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.
- You must set the SEC system initialization parameter to YES to enable security.
- You must have valid IBM 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 IBM 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 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 Group is set to the name of your CICS group.
   - The URM is set to point to the default IPIC autoinstall user program, DFHISAIP.
   - 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.

```
CEDA  DEFine TCpipservice( SRV50889 )
 TCpipservice   : SRV50889
 GROup          : HOLLTCPA
 DEscription  ==> IPIC AUTOINSTALL
 Urm          ==> DFHISAIP
 POrtnumber   ==> 50889            1-65535
 STatus       ==> Open             Open | Closed
 PROtocol     ==> IPIC             IIop | Http | Eci | User | IPic
 TRansaction  ==> CISS
 Backlog      ==> 00010            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, DFHISAIP.

     For example:

```
CEMT INQ TCPIPSERVICE
STATUS: RESULTS - OVERTYPE TO MODIFY
Tcpips(SRV50889) Ope Por(50889) Ipic Nos Tra(CISS)
Con(00000) Bac( 00010 ) Max( 000000 ) Urm( DFHISAIP )
```

**Note:** You can configure CICS resources using the CICS Explorer , see the CICS Explorer information in the CICS TS Information Center for more information.

## Testing your scenario

To test that your scenario is configured correctly, start the CICS Transaction Gateway and use the CICS TG Java sample EciB2 to call CICS server program EC01.

1. To test your scenario, 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 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 : null
   Password : null
   Data Conversion : ASCII
   Commarea        : null
   Commarea length : 18

   Number of programs given : 1
     [0] : EC01

   Connect to Gateway

   Successfully created JavaGateway

   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 DFHISAIP and template
   (NONE) after a connection request was received on tcpipservice SRV50889
   from host 1.23.456.789
   ```

   where `00000006` is the name of the IPCONN automatically generated by the autoinstall template, DFHISAIP.

   If you issue the command CEMT INQ IPCONN, the output is as follows:

   ```
   CEMT INQ IPCONN
     STATUS:  RESULTS - OVERTYPE TO MODIFY
      Ipc(00000006) App(00000006) Net(GBIBMIYA) Ins Acq Nos
         Rece(100) Sen(000) Tcp(SRV50889)
   ```

## 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 .... IPCONN APPL with applid MYQUAL.MYAPPL
autoinstalled successfully using autoinstall user program DFHISAIP
and template NONE after a connection request was received on
tcpipservice SRV50889 from host 1.23.456.789

DFHIS2001 .... Client session from applid MYAPPL accepted for IPCONN
APPL.
```

By default, the user replaceable module DFHISAIP 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), 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)
```

# Chapter 56. Configuring workload management using a CICS request exit (SC09)

This scenario shows how to configure workload management using a CICS request exit to select a server.

**Note:** This example uses CICS Transaction Gateway connecting to CICS Transaction Server V5.3 over IPIC and IBM WebSphere Application Server V8.0. For the minimum requirements, see "Prerequisites" on page 302.

This scenario uses the default name `ctg.ini` for the configuration file.

The figure shows workload from a connection factory in IBM WebSphere Application Server being served by a Gateway daemon. Any supported JEE application server can be used in this scenario.

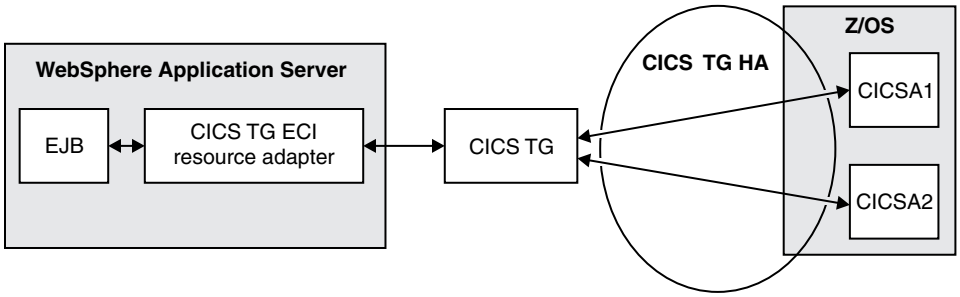



*Figure 14. Transactions over IPIC in a high availability scenario*

The Gateway daemon is connected to CICS servers CICSA1 and CICSA2 using the IPIC protocol. The work in CICS will be handled by one of CICSA1 or CICSA2. In this scenario, the Gateway daemon is configured for dynamic server selection using a CICS request exit. Dynamic server selection is performed at the start of each new transaction and manages the associated transactional affinity with the selected CICS server, for the life of the transaction.

**Note:** The connection factory definition in IBM WebSphere Application Server does not need to contain details of the actual CICS servers.

Follow the step-by-step instructions in this scenario to implement workload management using a CICS request exit.

*Table 27. Values used in this scenario*

| Comp-onent | Property | Where set | Details |
|---|---|---|---|
| CICS TG | Gateway daemon configuration | ctg.ini | cicsrequestexit=com.ibm.ctg.samples.ha.RoundRobinExit<br>protocol@tcp.handler=com.ibm.ctg.server.TCPHandler<br>protocol@tcp.parameters=port=2006; |
| CICS TG | IPIC connection to CICSA1 | ctg.ini | name=CICSA1<br>hostname=server.ibm.com<br>port=4149 |

*Table 27. Values used in this scenario (continued)*

| Comp-onent | Property | Where set | Details |
|---|---|---|---|
| CICS TG | IPIC connection to CICSA2 | ctg.ini | name=CICSA2<br>hostname=server.ibm.com<br>port=4150 |
| CICS TG | Round robin | ha.ini | CICSL1=CICSA1,CICSA2 |
| CICS TG | CLASSPATH | Environment variables | The location on the file system of the CA1T CICS request exit JAR file. |
| CICS TG | CTG_HACONFIG | Environment variables | The location on the file system of the round robin CICS request exit policy file. |
| CICS TS CICSA1 | TCPIPService for IPIC connection | Using CEDA on CICSA1 | name=IPIC4149<br>protocol=IPIC<br>port=4149 |
| CICS TS CICSA2 | TCPIPService for IPIC connection | Using CEDA on CICSA2 | name=IPIC4150<br>protocol=IPIC<br>port=4150 |
| WAS | CICS TG ECI resource adapter connection factory | J2C connection factories | name=ECI-CICSTG |
| WAS | CICS TG ECI resource adapter connection factory | J2C connection factories | JNDI Name=eis/CICSTG |
| WAS | CICS TG ECI resource adapter connection factory | CICS TG ECI resource adapter connection factory connection pool properties | Maximum connections=100 |
| WAS | Connection factory details | CICS TG ECI resource adapter connection factory custom properties | ConnectionURL=tcp://server.ibm.com<br>PortNumber=2006<br>ServerName=CICSL1 |

The sample configuration file and High Availability policy file for this scenario are provided in the directory `<install path>/samples/scenarios/sc09`:

- `ctg.ini`
- `ha.ini`

## Prerequisites

The prerequisites for CICS Transaction Gateway, CICS Transaction Server and IBM WebSphere Application Server.

Here are the system requirements for CICS Transaction Server for IBM z/OS:

- The server 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 TCPIP 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.

Here are the system requirements for CICS Transaction Gateway:

- CICS Transaction Gateway must be installed.
- The SupportPac CA1T "High Availability exit samples" must be available in the file system where the product was installed. For more information see `http://www-01.ibm.com/support/docview.wss?rs=1083&uid=swg27007241`.

Here are the system requirements for IBM WebSphere Application Server:

- The Installation Verification Test (IVT) that you run to test this scenario uses the ECI resource adapter, cicseci.rar. The resource adapter must be downloaded onto the machine that runs the Web browser you use for accessing the IBM WebSphere Application Server Integrated Solutions Console. The resource adapter will be installed as part of this scenario.

To test the scenario works successfully run the IVT. For more information see "JCA resource adapter installation verification test (IVT)" on page 251.

## Configuring CICS Transaction Gateway for high availability

To configure CICS Transaction Gateway for high availability, you need to create a number of configuration files.

The configuration file and the policy file are installed under the directory `<install_path>/samples/scenarios/sc09`.

The configuration files used to define a gateway for high availability are:

- A CICS Transaction Gateway configuration file.
- A High Availability policy file for use with the round robin CICS request exit.

### CICS Transaction Gateway configuration file

Create a CICS Transaction Gateway configuration file `ctg.ini` using the values suggested in the table Chapter 56, "Configuring workload management using a CICS request exit (SC09)," on page 301:

You must define server definitions for the Gateway daemon to communicate with CICS over IPIC in remote mode. To define the CICS server definitions for the Gateway daemon. Edit the IPICSERVER sections within the ctg.ini file:

- Set HOSTNAME to the name of the IBM z/OS machine that hosts your CICS server.
- Set PORT to the port number that your CICS server uses to listen for incoming IPIC requests.

For example:

```
SECTION IPICSERVER = CICSA1
    DESCRIPTION=IPIC connection to CICSA1
    HOSTNAME=server.ibm.com
    PORT=4149
ENDSECTION

SECTION IPICSERVER = CICSA2
    DESCRIPTION=IPIC connection to CICSA2
    HOSTNAME=server.ibm.com
    PORT=4150
ENDSECTION
```

### Configure round robin CICS request exit

Set the environment variable to point to the round robin policy file:

1. Add the CA1T SupportPac jar to the CLASSPATH.
2. Copy the sample `ha.ini` file to the configuration settings folder.
3. Set the environment variable *CTG_HACONFIG* to the location of the `ha.ini` file.

## 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).

For this scenario, the two CICS regions require separate TCPIPService definitions, using the port numbers defined in the table of values in Chapter 56, "Configuring workload management using a CICS request exit (SC09)," on page 301.

1. Use CEDA to define a TCPIPService; for example, SRV4149. The important values are:
   - The URM is set to point to your compiled IPCONN autoinstall user program.
   - The port numbers are 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( SRV4149 )
    TCpipservice   : SRV4149
    GROup          : HOLLTCPA
    DEscription  ==>
    Urm          ==> DFHISCIP
    POrtnumber   ==> [port number]    1-65535
    STatus       ==> Open              Open | Closed
    PROtocol     ==> IPIC              IIop | Http | Eci | User | IPic
    TRansaction  ==> CISS
    Backlog      ==> 00010             0-32767
    TSqprefix    ==>
    Ipaddress    ==>
    SOcketclose  ==> No                 No | 0-240000 (HHMMSS)
    Maxdatalen   ==>                    3-524288
   ```

2. Follow this pattern to create a TCPIPService definition for each CICS region. Allocate different port numbers to each definition. For example, 4149 for CICSA1 4150 for CICSA2, etc.
3. Install the CEDA definitions.
4. Check that the TCPIPService definitions are 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 the IPCONN autoinstall program that you modified.

     For example:

```
CEMT INQ TCPIPSERVICE
STATUS: RESULTS - OVERTYPE TO MODIFY
Tcpips(SRV4149) Ope Por(4149) Ipic Nos Tra(CISS)
Con(00000) Bac( 00010 ) Max( 000000 ) Urm( DFHISCIP )
```

## Configuring WebSphere Application Server

Use the WebSphere Application Server Integrated Solutions Console (the "admin console") to install and configure the CICS ECI resource adapter (cicseci.rar ).

To install and configure the CICS ECI resource adapter, you complete these key tasks:

1. Install the ECI resource adapter
2. Create a connection factory
3. Configure a connection factory

### Step 1. Install the ECI resource adapter

Start the admin console and select **Resource Adapters** from the **Resources > Resource Adapters** section of the navigation menu. Click **Install RAR** shown in Figure 15.

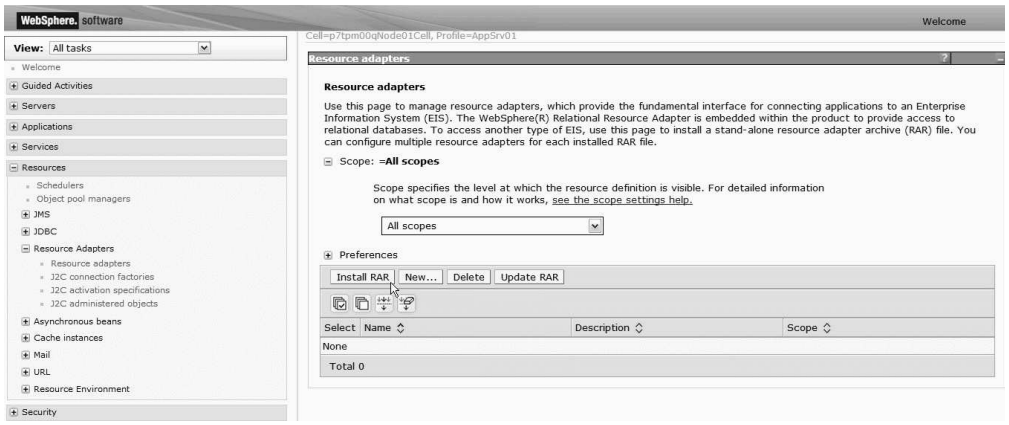



*Figure 15. Installing a new resource adapter*

From the **Install RAR file** dialog shown in Figure 16 on page 306, click **Browse** for the Local file system. Select the CICS ECI resource adapter `cicseci.rar`. Take note of the scope; this choice limits the scope of later connection factory definitions. In this scenario, the scope is *Node*.

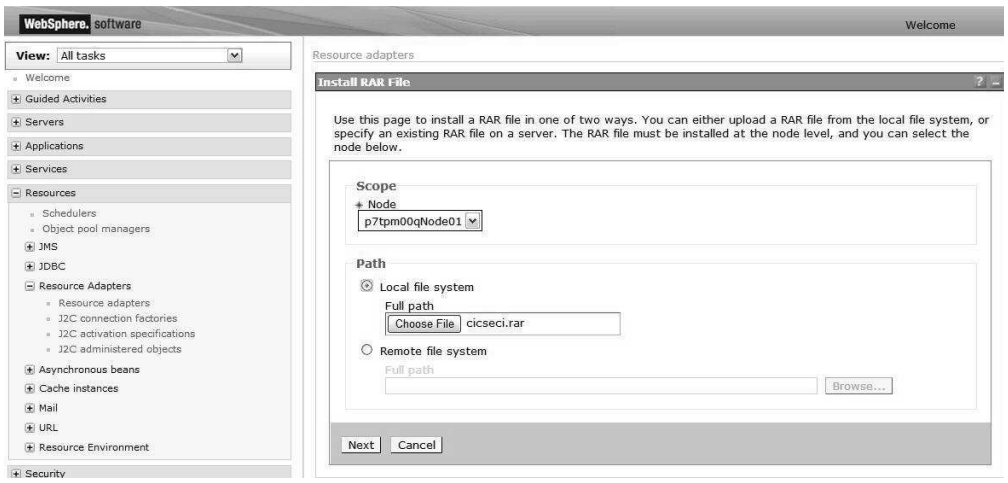



*Figure 16. Selecting the ECI RAR file for installation*

Click **Next**. The resource adapter **General Properties** dialog shown in Figure 17 on page 307 contains a predefined name and description for the CICS ECI resource adapter.

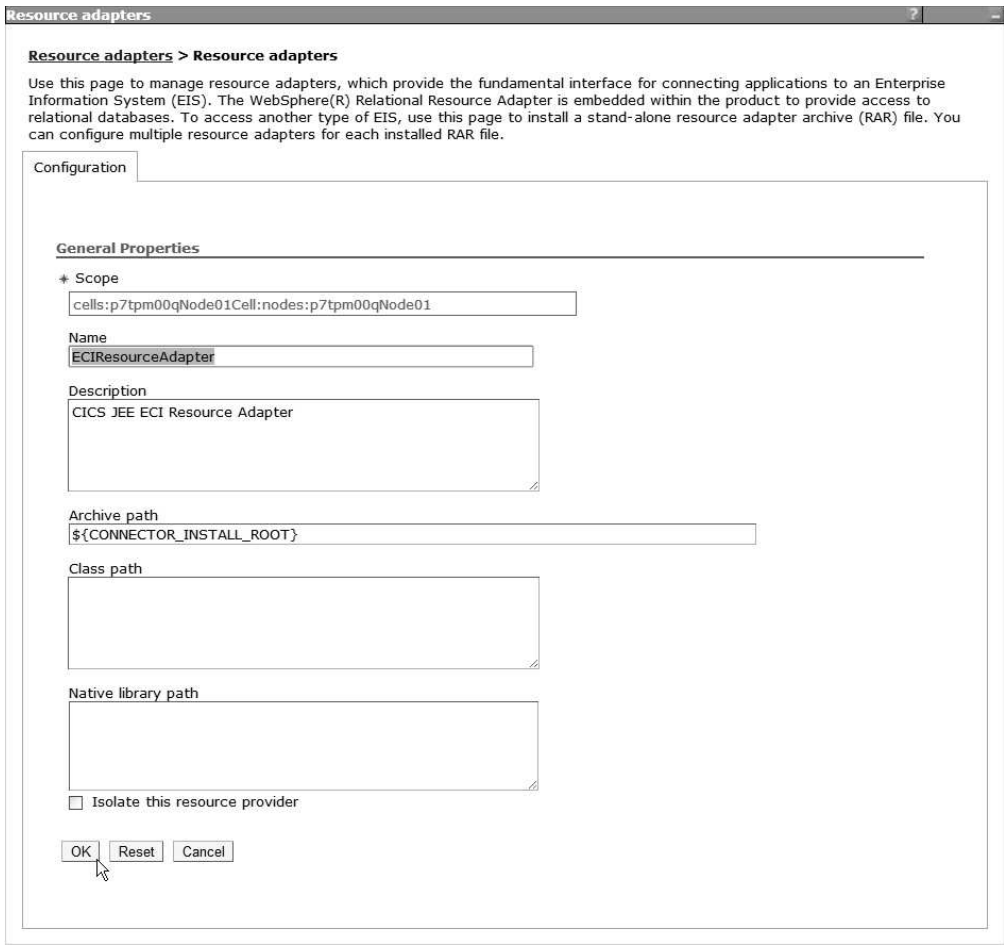


*Figure 17. Defining the RAR general properties*

Because this scenario implements the remote mode topology, you do not have to configure the native library path. Accept the default settings and click **OK**.

The admin console returns you to the **Resource adapters** dialog shown in Figure 18, and prompts you to save the changes. Note that the new resource adapter `ECIResourceAdapter` is now visible in the table of available resource adapters, with the scope matching the node.

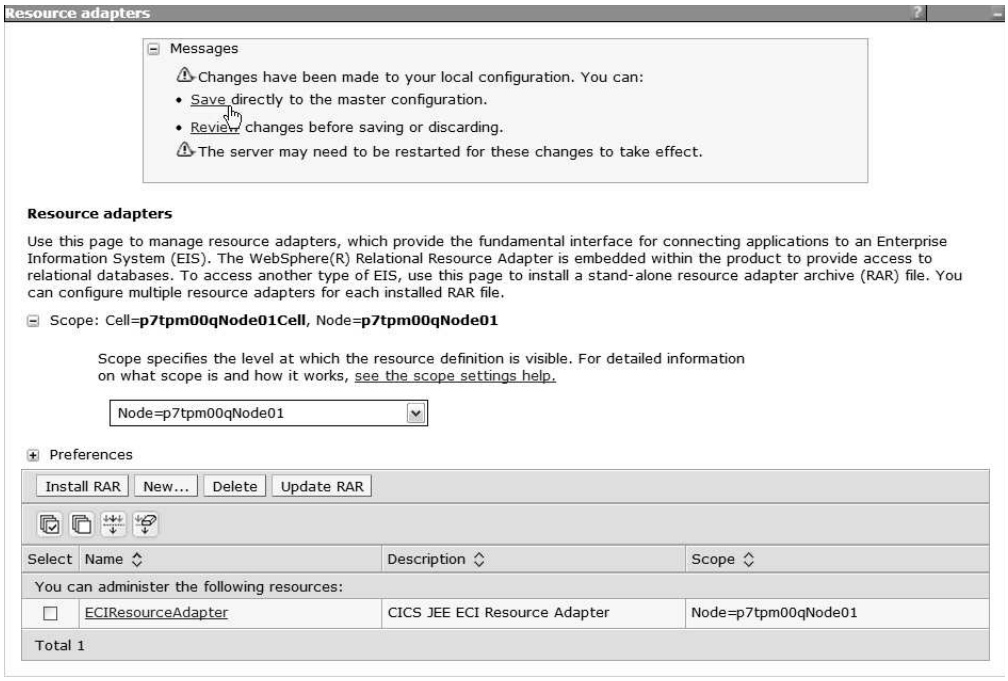



*Figure 18. Saving the configuration changes for the new resource adapter*

Save the changes to the master configuration.

## Step 2. Create a connection factory

Select **J2C connection factories** from the **Resources > Resource Adapters** section of the navigation menu. Firstly, you must ensure that the scope is correctly selected as shown in Figure 19 on page 308. For this particular scenario, the resource adapter is installed within the **Node** scope. If the scope is set incorrectly, attempts to create a new connection factory will fail.

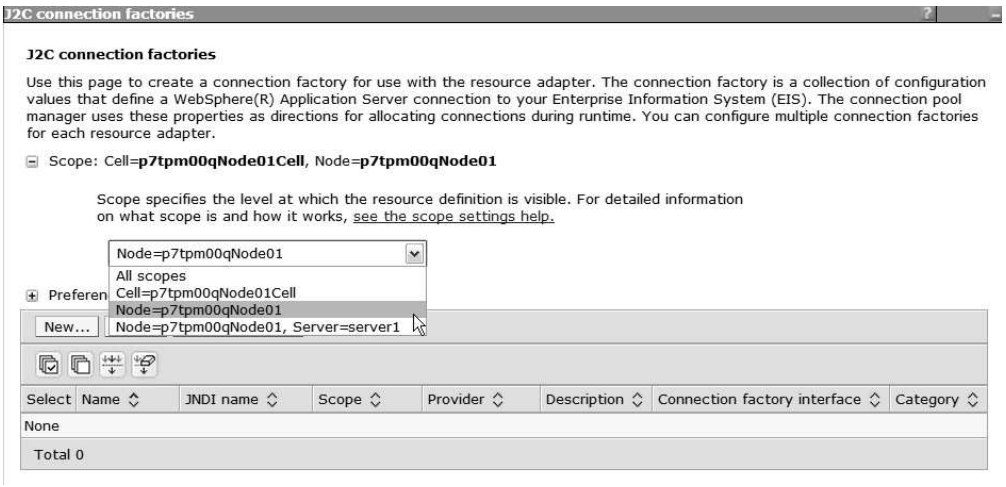



*Figure 19. Setting the scope for the new connection factory*

Click **New** to create a new connection factory using the CICS ECI resource adapter.

The J2C connection factory **General Properties** screen allows you to define the name, JNDI name and description for the new connection factory. The required values are provided in the table of values in Chapter 56, "Configuring workload management using a CICS request exit (SC09)," on page 301, and are shown in the following screen capture.

For the JEE applications, the important value here is the JNDI name. This allocates a single symbolic name to access CICS, and masks the detail of the underlying Gateway daemon and CICS topology.

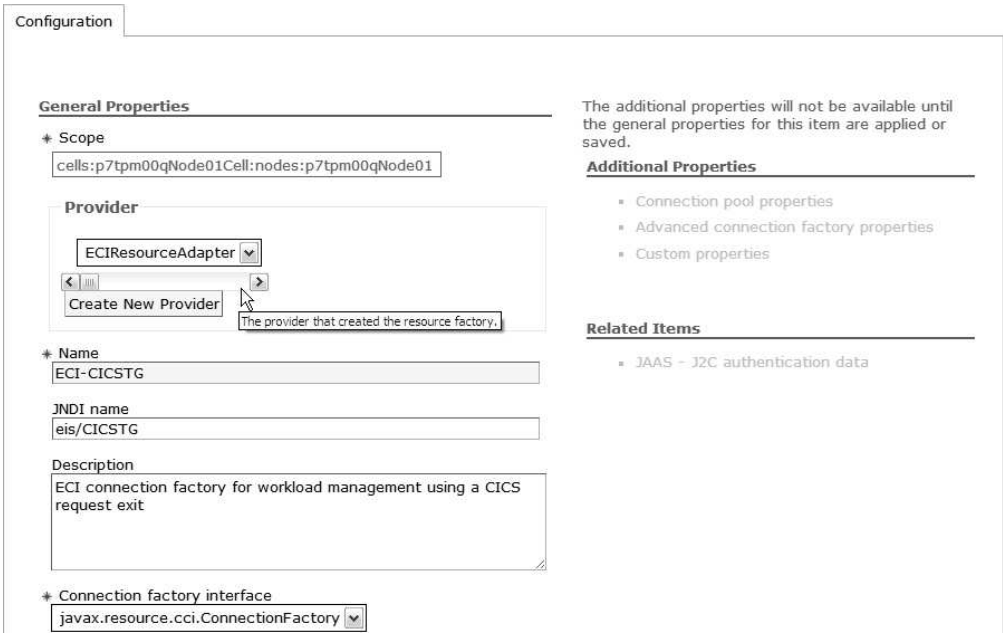



*Figure 20. Defining the new connection factory general properties*

**Note:** This dialog also contains configurable security settings for authentication but, because the scenario does not cover security, all security fields retain their default values.

At the bottom of this dialog, click **OK**. The admin console returns you to the **J2C connection factories** dialog. The new connection factory is included in the table as shown in Figure 21.

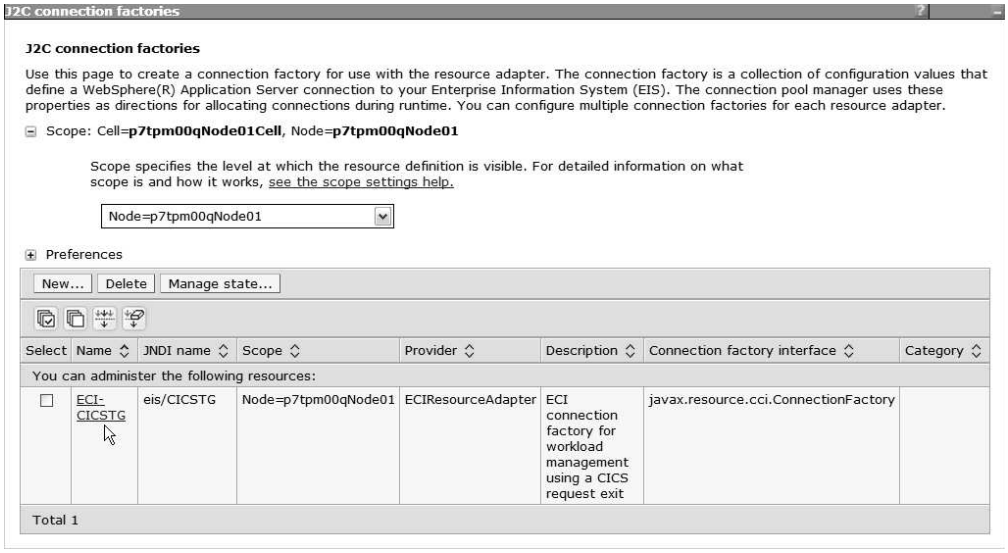



*Figure 21. The new J2C connection factory*

If the admin console prompts you to save the changes to the master configuration, you do not have to save at this point because the configuration task is not yet complete.

## Step 3. Configure a connection factory

In the **J2C connection factories** dialog, select the new connection factory ECI-CICSTG from the table of connection factories shown in Figure 21.

The admin console returns you to the **J2C connection factories - General Properties** dialog for the connection factory ECI-CICSTG. From the **Additional properties** section, select **Connection pool properties**. The admin console displays the **Connection pool properties** configuration dialog shown in Figure 22 on page 310.

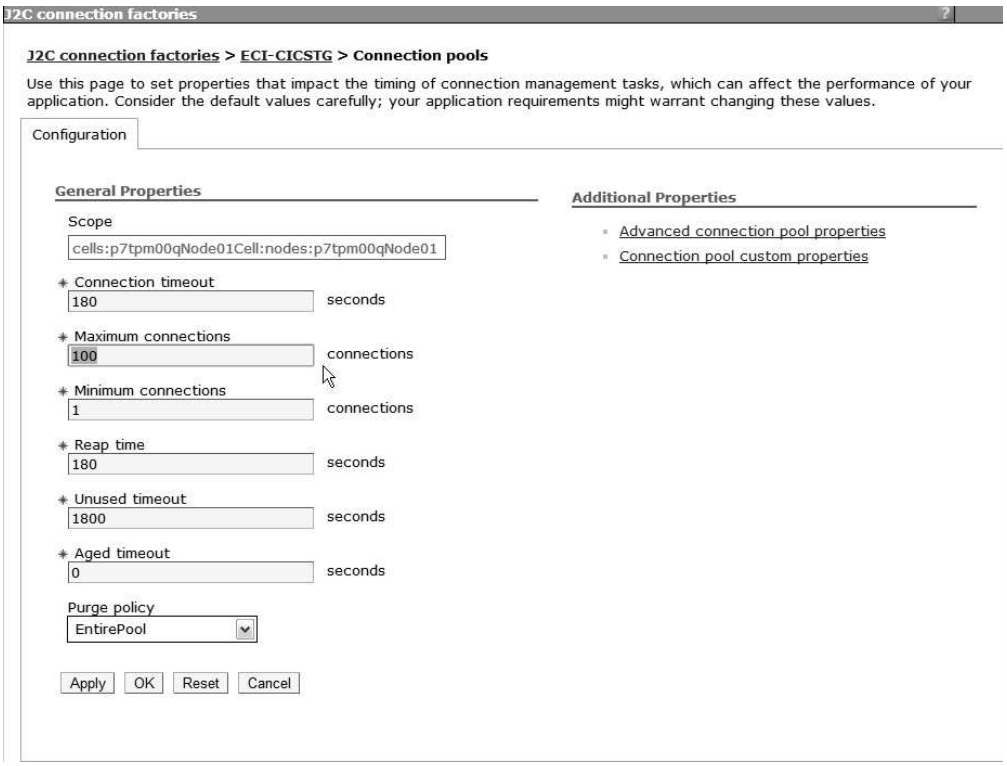


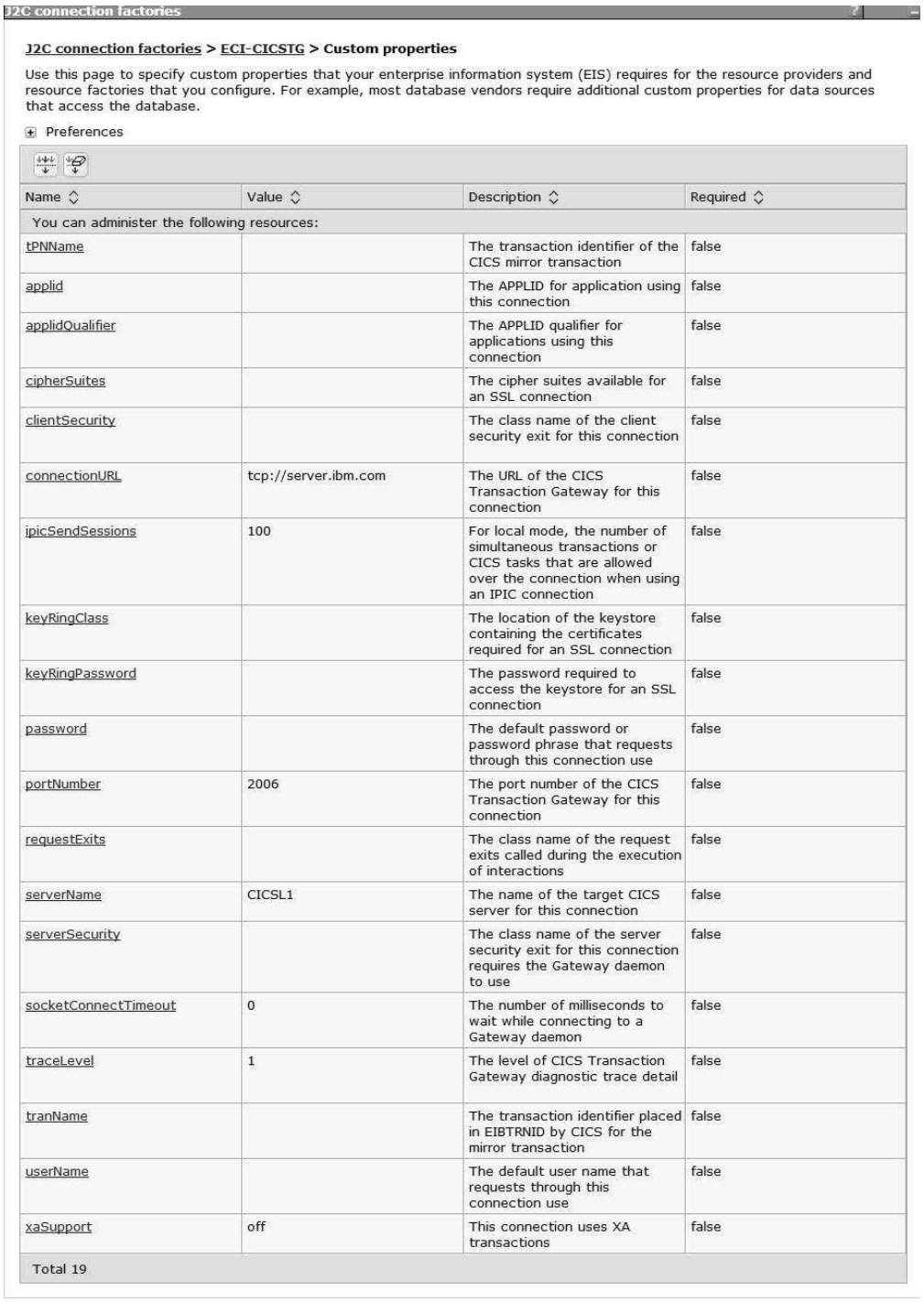

*Figure 22. Configuring the connection pools*

Change the value of *Maximum connections* to 100 to match the default MAXCONNECT value of 100 which the Gateway is configured to use.

For the remaining connection pool properties, use the default values. Click **OK**. The admin console returns you to the **J2C connection factories - General properties** dialog.

If the admin console prompts you to save the changes to the master configuration, you do not have to save at this point because the configuration task is not yet complete. The final part of the configuration task provides the connection factory with the details required to connect to a Gateway daemon and CICS.

From the **J2C connection factories - General properties** dialog, **Additional properties** section, select **Custom properties**. The admin console displays the **Custom properties** dialog shown in Figure 23 on page 311.



J2C connection factories

J2C connection factories > ECI-CICSTG > Custom properties

Use this page to specify custom properties that your enterprise information system (EIS) requires for the resource providers and resource factories that you configure. For example, most database vendors require additional custom properties for data sources that access the database.

Preferences

| Name | Value | Description | Required |
|---|---|---|---|
| You can administer the following resources: | | | |
| tPNName | | The transaction identifier of the CICS mirror transaction | false |
| applid | | The APPLID for application using this connection | false |
| applidQualifier | | The APPLID qualifier for applications using this connection | false |
| cipherSuites | | The cipher suites available for an SSL connection | false |
| clientSecurity | | The class name of the client security exit for this connection | false |
| connectionURL | tcp://server.ibm.com | The URL of the CICS Transaction Gateway for this connection | false |
| ipicSendSessions | 100 | For local mode, the number of simultaneous transactions or CICS tasks that are allowed over the connection when using an IPIC connection | false |
| keyRingClass | | The location of the keystore containing the certificates required for an SSL connection | false |
| keyRingPassword | | The password required to access the keystore for an SSL connection | false |
| password | | The default password or password phrase that requests through this connection use | false |
| portNumber | 2006 | The port number of the CICS Transaction Gateway for this connection | false |
| requestExits | | The class name of the request exits called during the execution of interactions | false |
| serverName | CICSL1 | The name of the target CICS server for this connection | false |
| serverSecurity | | The class name of the server security exit for this connection requires the Gateway daemon to use | false |
| socketConnectTimeout | 0 | The number of milliseconds to wait while connecting to a Gateway daemon | false |
| traceLevel | 1 | The level of CICS Transaction Gateway diagnostic trace detail | false |
| tranName | | The transaction identifier placed in EIBTRNID by CICS for the mirror transaction | false |
| userName | | The default user name that requests through this connection use | false |
| xaSupport | off | This connection uses XA transactions | false |
| Total 19 | | | |

*Figure 23. Configuring the connection factory custom properties*

Set each of the following custom properties individually, by clicking on the name:

ConnectionURL

PortNumber

ServerName

The required values for these properties are provided in the table in Chapter 56, “Configuring workload management using a CICS request exit (SC09),” on page 301; the default values for all other properties are acceptable. When you have set the required custom properties, save all changes to the master configuration.

## Testing the scenario

To test the workload management using a CICS request exit scenario, you configure, then run the installation verification test (IVT) supplied with CICS Transaction Gateway.

### Step 1. Configure the IVT

Install the IVT application ECIIVT.ear. For more information see “JCA resource adapter installation verification test (IVT)” on page 251.

Configure the IVT application to use the gateway, by specifying the resource reference.

Select **WebSphere enterprise applications** from the **Applications > Application types** section of the navigation menu. From the **Enterprise Applications** dialog, **Preferences** section, click **ECIIVT** from the table of applications.

From the **Enterprise Applications** configuration panel for ECIIVT, **References** section, click **Resource references**.

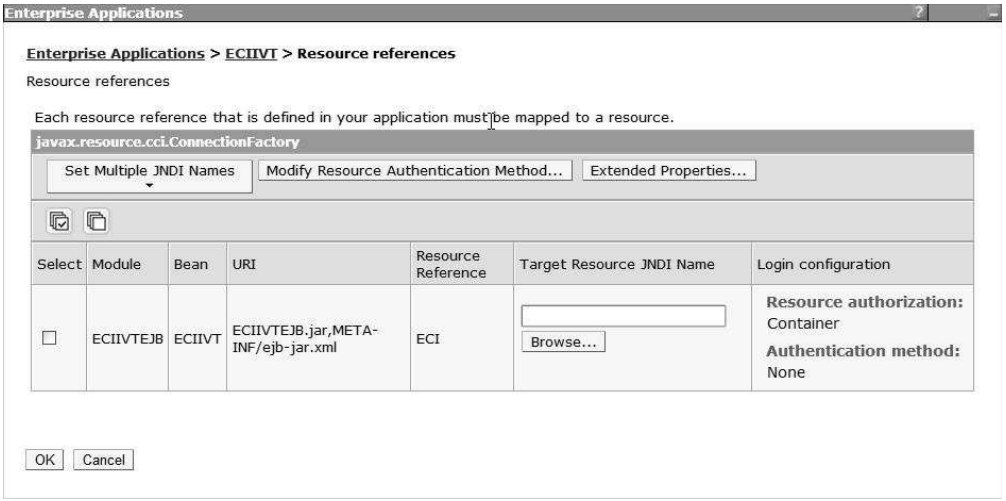



*Figure 24. ECI IVT resource references*

The **Resource references** dialog shown in Figure 1 allows you to specify the name of the JNDI (Java Naming Directory Interface) to be used by the application. Click **Browse** to display the **Available resources** dialog shown in Figure 2.

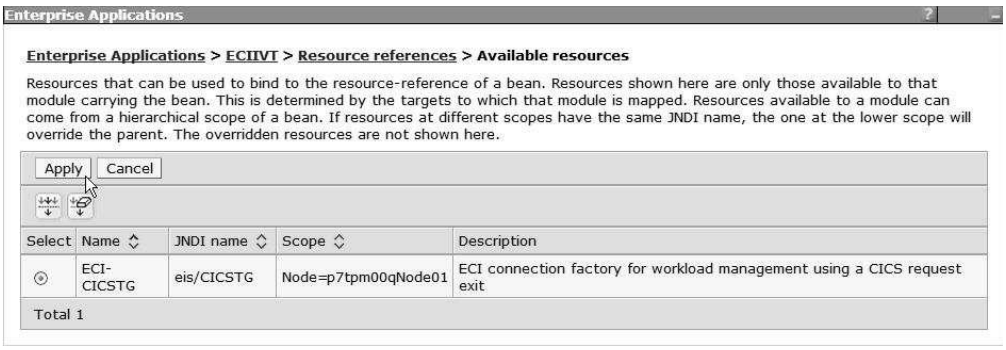



*Figure 25. Available resources*

Select ECI-CICSTG resource and click **Apply**. The admin console returns you to the **Resource references** dialog. The **Target Resource JNDI Name** now has the value *eis/CICSTG* shown in Figure 3.

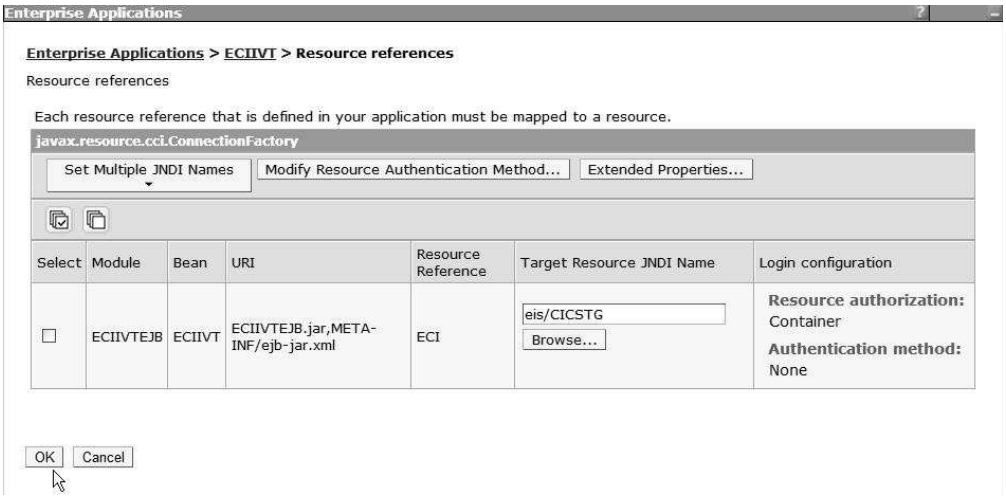



*Figure 26. Configured ECI IVT resource references*

You have now configured the ECIIVT application to use the gateway. Save the changes to the master configuration.

## Step 2. Run the IVT

Start the IVT application in IBM WebSphere Application Server and start the gateway daemon.

Run the ECI IVT several times to send ECI requests to CICS, using different CICS servers each time. If at least one of the CICSA1, CICSA2 servers are available, the request will succeed.

For more information about how to run the IVT see

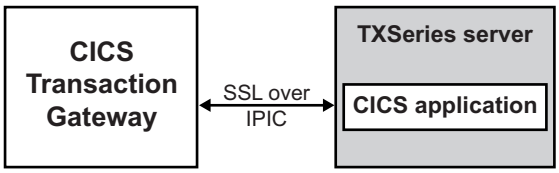
`ftp://ftp.software.ibm.com/software/htp/cics/support/supportpacs/individual/ch91.pdf`

# Chapter 57. Configuring SSL between CICS TG and IBM TXSeries (SC10)

This scenario shows you how to configure SSL security on an IPIC connection between CICS Transaction Gateway and IBM TXSeries for Multiplatforms V8.1.

IBM TXSeries for Multiplatforms v8.1 can access keystores of type `CMS` or `PKCS12` and uses the IBM Global Security Kit to generate keystores and manage certificates. CICS Transaction Gateway can only access keystores of type `jks` (java keystore) using the iKeyman Java tool to generate keystores and manage certificates. The following scenario uses the fact that, both iKeyman and the IBM Global Security Kit can access `PKCS12` type keystores, to transfer certificates between the two systems.

The following figure shows the topology used in this scenario:




## Prerequisites for the SSL scenario

Before you can complete this scenario, you must ensure that the system requirements for IBM TXSeries and CICS Transaction Gateway are satisfied.

**Note:** This scenario is only applicable to IBM TXSeries.

- IBM TXSeries for Multiplatforms V8.1 uses the IBM Global Security Kit to generate keystores and manage certificates.
- The version of the IBM global security kit installed must be as specified in the IBM TXSeries for Multiplatforms documentation.

CICS Transaction Gateway:

- CICS Transaction Gateway must be correctly 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 you must install the sample CICS COBOL programs EC01, EC03 on the CICS server.

For information about the samples see CICS server applications.

## Configuring the connection

To complete this task you use the IBM Global Security Kit commands to create a Key Database, SSL certificate and public certificate; define the service name and port; configure the IBM TXSeries listener to use them; transfer and add the public certificate to CICS TG machine and then configure the CICS TG to use it.

1. Create a Key Database, SSL certificate and public certificate.

a. Use the IBM Global Security Kit command to create a Key Database. You can create a cms or pkcs12 type of key database. Ensure the file permissions on the key database files allow the IBM TXSeries server to read from the key database. For example:

```
gsk8capicmd -keydb -create -db /home/ctg/ssl/ctgkey.kdb -pw password -type
 cms
```

b. Use the IBM Global Security Kit command to either add or create a certificate in the store. For example, to create an new certificate:

```
gsk8capicmd -cert -create -db /home/ctg/ssl/ctgkey.kdb -pw password -label
 ctgcert -dn CN=txseries1.company.com
```

c. Use the IBM Global Security Kit command export the public certificate. For example:

```
gsk8capicmd -cert -export -db  /home/ctg/ssl/ctgkey.kdb -pw password -label
 ctgcert  -type cms -target  /home/ctg/ssl/txcert.p12 -target_pw password2
 -target_type pkcs12
```

2. Configure the service name and port used in the /etc/services file
   - The TCP Service name must be mapped to a port in the etc/services file add an entry of the format

```
31031IX3        31031/tcp               # SSL TX IPIC  TX3
```

3. Modify the following IBM TXSeries Listener Definition fields expanding any groups as necessary:

| Field | Description |
|---|---|
| Listener Name | Required field. Enter a name of your choice. In this example, we chose the same name as the service name: 31031IX3. |
| Activate resource at cold start? | Set to Yes. |
| TCP adapter address | The IP address of this IBM TXSeries machine. For example: txseries1.company.com. |
| TCP service name | The TCP Service name must be mapped to a port in the etc/services file. Add an entry of the format: name number/tcp # comment. For example: 31031IX3. |
| Fully qualified pathname of the SSL keyring file | The fully qualified pathname of the Key Database created above: For example: /home/ctg/ssl/ctgkey.kdb. |
| Password of the SSL keyring file | Password of the Key Database created above. |
| Level of SSL authentication required | Set to Yes. |
| Level of SSL encryption required | Set to Yes. |
| Client's certificate name in keyring file | The label used to create or add the certificate to the Key Database: For example: ctgcert. |

4. Add the public certificate to the keystore on the CICS Transaction Gateway server machine.
   a. Transfer the new keystore, txcert.p12, to the CICS Transaction Gateway server machine using an FTP client.
   b. Start iKeyman and load the certificate file (txcert.p12)

c. Export the certificate `ctgcert` to the server keystore got from the `ctg.ini` file (`ServerKeyRing.jks`).

5. Configure CICS Transaction Gateway.
   - Configure the following parameters in the `ctg.ini` file.

```
SECTION PRODUCT
    KEYRING=c:\windows\temp\ssl\ServerKeyring.jks
ENDSECTION
SECTION IPICSERVER = IPSSETX
    SRVIDLETIMEOUT=0
    HOSTNAME=txseries1.company.com
    PORT=31031
    CONNECTTIMEOUT=0
    SRVRETRYINTERVAL=60
    TCPKEEPALIVE=Y
    SENDSESSIONS=20
    ECITIMEOUT=0
    SSL=Y
ENDSECTION
```

You have now configured an SSL link between CICS Transaction gateway and IBM TXSeries.

# Chapter 58. Configuring an existing CICS transaction as a JSON web service (SC11)

This scenario shows how to configure the CICS TG mobile support to pass a request from a JSON web services client application to the EC03 program on a CICS server and receive a response.

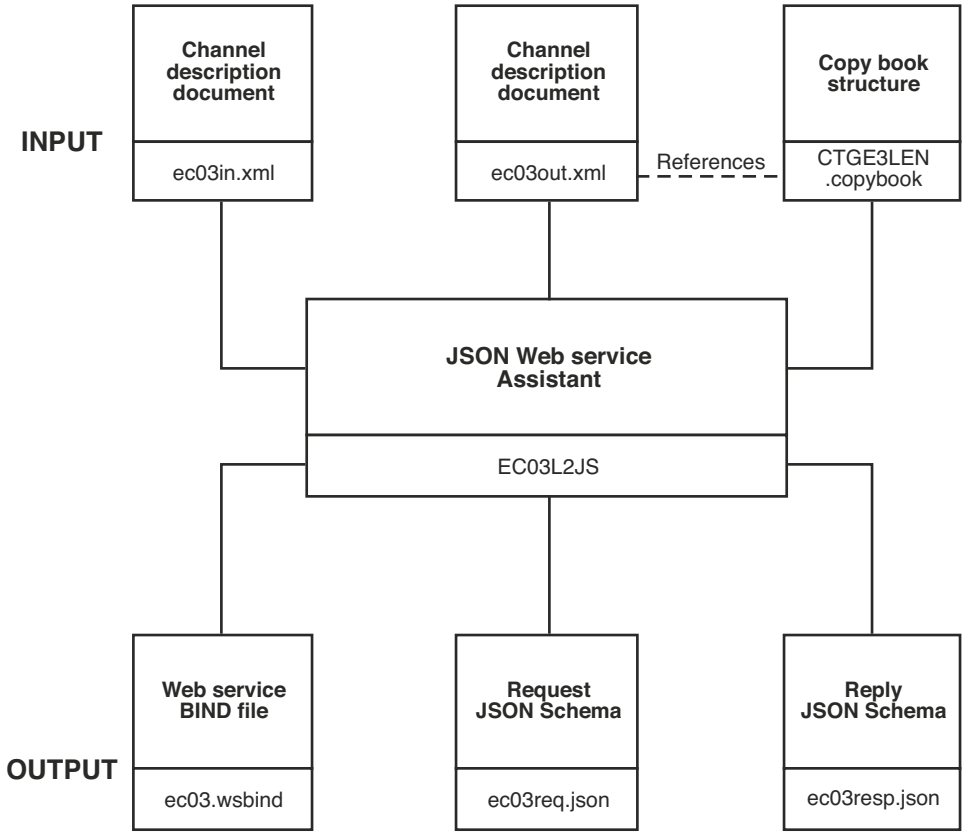
The following diagram shows the path that is taken by the request and response from a JSON web service client to the CICS TS EC03 sample program through CICS TG.


The files for this scenario are installed in the directory `<install path>/samples/scenarios/sc11`.

The files stored in `<install path>/samples/scenarios/sc11` are:

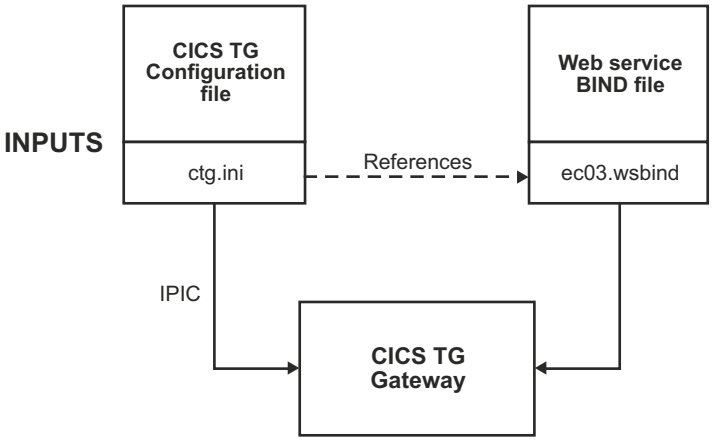
- `ec03in.xml` is the channel description document of the request sent to the CICS server program EC03.
- `ec03out.xml` is the channel description document of the reply from the CICS server program EC03.
- `ctgec03.ini` contains the relevant sections from the CICS TG configuration file required to implement this scenario.
- `ec03req.json` is the request json schema.
- `ec03resp.json` is the reply json schema.
- `ec03.wsbind` is the web service bind file.
- `CTGE3L2J.txt` is a parameter file to be used with the JSON web services assistant (ctgassist). This must be modified before use.
- `CTGE3LEN.copybook` is the COBOL copybook containing the language structure of the binary data passed in the BIT channel for the EC03 sample program.

This scenario shows you how to produce a web services bind file from two channel description documents (CDDs). The channel description documents describe the input and output of the EC03 CICS server sample that is included in `<install path>/samples/server/ec03.cpp`. The following diagram shows the inputs and outputs for the JSON web services assistant.




The bind file and JSON Schema that are produced are included so that you can configure a web service without going through the process of producing the bind file from CDDs. An extract from a CICS TG configuration file is also included to show the relevant sections of the CICS TG configuration file to configure a web service with this bind file.

The following diagram shows the relationship between the `ctg.ini` configuration file and the WSBind file




The JSON Schema files are provided for you to use in your client application.

## Prerequisites

The following resources are needed for this scenario:

- A CICS system running the CICS TG Server Sample EC03.
- A CICS TG system with an IPIC connection to the CICS system running the CICS TG Server Sample EC03.
- A REST Client application or browser plug-in to drive the web service.

## Producing the WSBind file

These steps describe how to create the WSBind file

### Procedure

1. Make a working copy of the JSON web services assistant parameter file `CTGE3L2J.txt` in a writable user-defined directory. In the following instructions, substitute the name of your working copy for `CTGE3L2J.txt`.
2. Make working copies of the channel description document `ec03in.xml` and `ec03out.xml` in a writable user-defined directory. In the following instructions, substitute the name of your working copy for `ec03in.xml` and `ec03out.xml`.
3. Make a working copy of the copybook for the CICS server program `CTGE3LEN.copybook` in a readable user-defined directory.
4. Edit the parameter file `CTGE3L2J.txt`:
   a. Replace the `<input path>` and `<output path>` tags with a path to the user-defined writable directory.
5. Edit the channel description document `ec03out.xml` to replace the `<install path>` tag with the name of the user-defined directory where you copied `CTGE3LEN.copybook`.
6. Run the assistant using: `ctgassist <path to>/CTGE3L2J.txt`.
7. Copy the WSBind file `ec03.wsbind` to a readable user-defined directory or use the default directory : `/var/cicscli` on any UNIX operating system or the directory that is specified by the value of the *CTG_DATA_PATH* environment variable on any Windows operating system.

### Results

The `ec03.wsbind` file, and the JSON Schema files `ec03req.json` and `ec03resp.json`, are produced and placed in the user-defined writable directory on the UNIX subsystem.

## Configuring HTTP

Enable the HTTP protocol to accept web service requests.

### Procedure

In the GATEWAY section of the `ctg.ini` configuration file, add the following subsection:

```
SUBSECTION HTTP
  port=1234
  bind=server.ibm.com
ENDSUBSECTION
```

## Configuring CICS TG for web services

Configure CICS TG to use the WSBind file for web services.

### Procedure

1. Edit the `ctgec03.ini` file:
   a. Replace the `<bind file directory>` tag with the directory that contains the bind file. You can also delete this tag to use the default directory.
   b. Replace the `<host name>` tag with the IP address or name of the host.
   c. Replace the `<server name>` tag with the IP address or name of the CICS server.
2. Merge the data in the `ctgec03.ini` file into your existing CICS TG configuration file (`ctg.ini`). The configuration file must include an IPICSERVER section that defines an IPIC connection to your CICS server.
3. Start CICS TG with the configuration file `ctg.ini`.

## Producing a client application from the JSON Schema

These steps show you how to use the JSON Schemas to define the JSON request and response payloads that are used by a client application to communicate with the sample EC03 program.

In the step "Producing the WSBind file" on page 321, the JSON web services assistant created the JSON Schema documents, `ec03req.json` and `ec03resp.json` that define the JSON data format of the request and response payloads.

JSON data is written as `name:value` pairs. A JSON object is one or more `name:value` pairs, separated by commas, and contained within braces.

For example: `{"forename":"John","lastname":"Smith"}`

### Example 1; Creating JSON request data to conform to the JSON Schema

The JSON request data must conform to the JSON Schema `ec03req.json`.

The schema document describes the JSON input format to the web service to drive the EC03 CICS transaction:

```
{
   "$schema":"http:\/\/json-schema.org\/draft-04\/schema#",
   "description":"Request schema for the EC03 JSON interface",
   "type":"object",      1
   "properties":{
      "EC03Operation":{      2
         "type":"object",      3
         "properties":{
            "INPUTDATA":{         4
               "type":"string"      5
            }
         },
         "required":[
            "INPUTDATA"
         ]
      }
   },
```

```
   "required":[
      "EC03Operation"
   ]
}
```

The JSON Schema shows that the JSON data must include a JSON object ( **1** ), with the name `"EC03Operation"` ( **2** ) whose value is also a JSON object ( **3** ). The nested JSON object has the name `"INPUTDATA"` ( **4** ) and a value of type `string` ( **5** ).

Therefore, the JSON request data is of the form

```
{"EC03Operation":{ "INPUTDATA": "testing data" }}
```

## Example 2: Creating JSON response data to conform to the JSON Schema

The JSON response data must conform to the JSON Schema `ec03resp.json`.

The JSON schema document describes the format of the JSON output of the web service that is returned from the EC03 CICS transaction:

```
{
   "$schema":"http:\/\/json-schema.org\/draft-04\/schema#",
   "description":"Response schema for the EC03 JSON interface",
   "type":"object",
   "properties":{
      "EC03OperationResponse":{
         "type":"object",
         "properties":{
            "INPUTDATALENGTH":{
               "type":"object",
               "properties":{
                  "inputlength":{
                     "type":"integer",
                     "maximum":2147483647,
                     "minimum":-2147483648
                  }
               },
               "required":[
                  "inputlength"
               ]
            },
            "OUTPUTMESSAGE":{
               "type":"string"
            },
            "CICSDATETIME":{
               "type":"string"
            }
         },
         "required":[
            "INPUTDATALENGTH",
            "OUTPUTMESSAGE",
            "CICSDATETIME"
         ]
      }
   },
   "required":[
      "EC03OperationResponse"
   ]
}
```

This example shows that the returned JSON payload must conform to the JSON Schema formats and be of the form:

```
{
   "EC03OperationResponse":
   {
      "CICSDATETIME": "18/03/2014 13:13:35",
      "OUTPUTMESSAGE": "Input data was: testing data",
      "INPUTDATALENGTH":
      {
          "inputlength": 12
      }
   }
}
```

# Testing the scenario

Testing of the web service on the CICS Transaction Gateway can be done with any REST Client application or the REST Client browser plug-in, or a IBM Worklight® HTTP adapter.

## Testing the scenario with a REST Client browser plug-in

Testing of the web service on the CICS Transaction Gateway can be done with any REST Client application or the REST Client browser plug-in.

### Procedure

1. Set the HTTP header **Content-Type** and **Accept** to `application/json`
2. Set the HTTP payload to `{"EC03Operation":{ "INPUTDATA": "testing data" }}`
3. Post the JSON payload to `http://<CICS TG host name>:2080/EC03`

### Results

The response headers should be:

```
Status Code: 200 OK
Content-Length: 150
Content-Type: application/json
Date: Wed, 02 Apr 2014 16:15:24 GMT
X-Powered-By: Servlet/3.0
```

The raw response body should be:

```
{"EC03OperationResponse":{"OUTPUTMESSAGE":"Input data was: testing data",
"INPUTDATALENGTH":{"inputlength":12},"CICSDATETIME":"02\/04\/2014 17:15:24"}}
```

## Testing the scenario with a IBM Worklight HTTP adapter

This procedure uses IBM Worklight Studio that is installed on Eclipse. The instructions apply to IBM Worklight Studio v6.1.

### Procedure

1. In a IBM Worklight project, create a new IBM Worklight HTTP adapter named `CtgDemoAdapter`.
2. Open the `CtgDemoAdapter.xml` file in the editor
3. Add a connections policy with the following settings:

| Parameter | Value |
|---|---|
| **Protocol** | `http` |
| **Domain** | `<cics tg host name>` |
| **Port** | `2080` |

4. Add a new procedure with the name `postEC03`.
5. Close and save `CtgDemoAdapter.xml`.
6. Open the `CtgDemoAdapter-impl.js` file in the editor and add the following function code:

```
function postEC03(ec03Data) {
   var request = '{"EC03Operation":{"INPUTDATA": ' + ec03Data +' }}';
   var input = {
      method : 'post',
      returnedContentType : 'json',
      path : 'EC03',
      headers : { 'Content-Type' : 'application/json'},
      body: {
         contentType: 'json',
         content: request.toString()
      }
   };
   return WL.Server.invokeHttp(input);
}
```

7. To send the request to CICS TG, right-click the **CtgDemoAdapter** adapter and select **Run As** > **Invoke Worklight Procedure**. In the Invoke Worklight Procedure dialog, set the following parameters:

| Parameter | Value |
|---|---|
| **Procedure name** | postEC03 |
| **Signature** | postEC03(ec03Data) |
| **Parameters** | 'Test data' |

## Results

A pane with the following response is displayed:

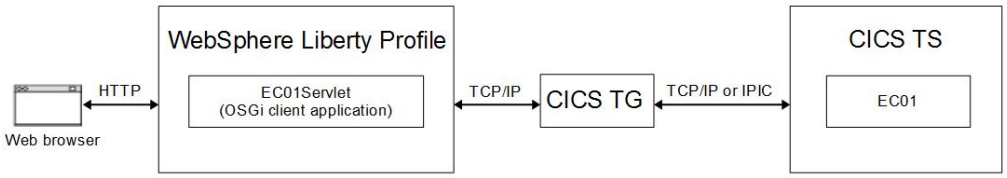
```
{
   "EC03OperationResponse": {
      "CICSDATETIME": "26\/03\/2014 12:54:03",
      "INPUTDATALENGTH": {
         "inputlength": 9
      },
      "OUTPUTMESSAGE": "Input data was: Test data"
   },
   "isSuccessful": true,
   "responseHeaders": {
      "Content-Length": "146",
      "Content-Type": "application\/json",
      "Date": "Wed, 26 Mar 2014 12:54:07 GMT",
      "X-Powered-By": "Servlet\/3.0"
   },
   "responseTime": 31,
   "statusCode": 200,
   "statusReason": "OK",
   "totalTime": 31
}
```

You can use the WL.Client.invokeProcedure function call from a IBM Worklight application to call this adapter.

# Chapter 59. Using the CICS TG API OSGi bundle (SC13)

This scenario shows how to connect to a remote Gateway daemon from a servlet running in IBM WebSphere Liberty Profile using the CICS TG API OSGi bundle.

The following figure shows the topology used in this scenario:




## Prerequisites

The prerequisites for IBM WebSphere Liberty Profile, CICS Transaction Server, and CICS Transaction Gateway.

The following products must be installed:

**System requirements for IBM WebSphere Liberty Profile**
IBM WebSphere Liberty Profile v8.5.5

**System requirements for the CICS Transaction Server for IBM z/OS servers**
If IPIC connections are being used, CICS Transaction Server V3.2 or later is required 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 TCPIP system initialization parameter to YES. To check the status of these services, use the CEMT INQ TCPIP command and check that the status is open.

The CICS server must have access to a TCP/IP stack that is running on the same LPAR.

**System requirements for CICS Transaction Gateway**
CICS Transaction Gateway must be installed.

You must have a SERVER or IPICSERVER section defined in the configuration file so that CICS Transaction Gateway can communicate with the CICS region. The server name must be defined as the DefaultServer value in the PRODUCT section of the configuration file.

## Configuring CICS TG to accept ECI requests

This scenario uses a TCP protocol handler to accept ECI requests.

In the CICS TG configuration file, within the SECTION GATEWAY, define a TCP protocol handler. For example:

```
SECTION GATEWAY
  protocol@tcp.handler=com.ibm.ctg.server.TCPHandler
  protocol@tcp.parameters=port=2006;
ENDSECTION
```

## Configuring the Liberty server

This scenario uses a servlet running in Websphere Liberty Profile.

### About this task

To configure the Liberty server perform the following steps:

### Procedure

1. Extract the CICS TG Client API bundle, `com.ibm.ctg.client-1.0.0.jar`, from the CICS TG SDK package, `cicstgsdk/api/java/runtime` to a directory on the local file system.
2. Add the following configuration to the `server.xml` file to add the bundle to the Liberty runtime:

   ```
   <bundleRepository>
     <fileset dir="/home/cicstg/bundles" scanInterval="10">
     </fileset>
   </bundleRepository>
   ```

   The **dir** parameter points to the location where the bundle was extracted to.
3. Download the sample application, `com.ibm.ctg.samples.osgi.eci.app_1.0.0.0.eba`,from the CICS TG SDK package, `cicstgsdk/api/java/samples` to a directory on the local file system..
4. Add the following configuration to the `server.xml` file to configure the required Liberty features:

   ```
   <featureManager>
     <feature>jsp-2.2</feature>
     <feature>blueprint-1.0</feature>
     <feature>wab-1.0</feature>
   </featureManager>
   ```
5. Add the following configuration to the `server.xml` file to add the application to the Liberty runtime:

   ```
   <osgiApplication id="com.ibm.ctg.samples.osgi.eci.app"
   location="/home/cicstg/com.ibm.ctg.samples.osgi.eci.app_1.0.0.0.eba"
   name="com.ibm.ctg.samples.osgi.eci.app"/>
   ```

   The **location** parameter is the where the application file was stored.
6. Set **JAVA_HOME** to point to a Java 7 installation.

## Testing the application

Follow these steps to test the application for Scenario 13

### Procedure

1. Start the CICS Transaction Gateway
2. Start the IBM WebSphere Liberty Profile server instance
3. In a web browser go to the address:

   ```
   http://<hostname>:<port>/com.ibm.ctg.samples.osgi.eci/EC01Servlet
   ```

   where `<hostname>` is the hostname of the machine where the IBM WebSphere Liberty Profile is installed and `<port>` is the httpPort defined in the httpEndpoint section of the `server.xml` file.
4. Enter the hostname and port of the Gateway daemon into the form and click the button.

5. If the date and time from CICS are displayed in the browser, then the test was successful. If the test failed, an error message is displayed.

# Part 8. Operating

When operating CICS Transaction Gateway on UNIX and Linux, it is important to start and stop the components in the correct sequence, so that transactions can complete successfully.

The CICS Transaction Gateway configuration file is processed during CICS Transaction Gateway initialization. If changes are made to the configuration file whilst CICS Transaction Gateway is running, the changes will have no effect until CICS Transaction Gateway is restarted.

# Chapter 60. Starting CICS Transaction Gateway on UNIX and Linux

You can start CICS Transaction Gateway in console mode or as a background task. The Client daemon always runs as a background task.

## Starting the Gateway daemon in console mode

On UNIX and Linux, you can start the Gateway daemon in console mode where you can specify options to override parameter values specified in the configuration file `ctg.ini`, or to override default JVM settings.

To start the Gateway daemon in console mode use the **ctgstart** command. For more information see the "ctgstart command reference" on page 369

If the Client daemon is not already started, it is started automatically by the **ctgstart** command before the Gateway daemon is started.

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 161.

## Running the Gateway daemon as a background process

On UNIX and Linux you can run the Gateway daemon as a background process with preset options defined in `ctg.ini` or you can specify override options using the **ctgd** command.

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.
- Start and stop the Gateway daemon when the operating system starts and stops.

To start the Gateway daemon as a background process use the **ctgd** command. For more information about **ctgd** see the "ctgd command reference" on page 364. **ctgd** must be run as the root user. To configure **ctgd** to be called during the startup and shutdown of your 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.

If the Client daemon is not already started, it is started automatically by the **ctgd** command before the Gateway daemon is started. The Client daemon is not stopped when the Gateway daemon is stopped.

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 161. The **ctgd** command does not use the port specified in the Gateway daemon configuration file.

### 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;
```

For information about Gateway daemon configuration settings, see "Gateway daemon logging" on page 161.

### Defining the ctgd.conf configuration file

Create a valid `ctgd.conf` configuration file. Copy the sample file `<install_path>/samples/configuration/ctgdsamp.conf` and edit the file copy. For information about the configuration file parameters, see "ctgd command reference" on page 364.

The default location and name for the **ctgd** configuration file is `<install path>/bin/ctgd.conf`.

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

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

### Configuring the Gateway daemon log

The default destination for the Gateway daemon log is `file`. This destination must be used when the Gateway daemon is run in the background. For information about Gateway daemon configuration settings, see "Gateway daemon logging" on page 161.

## Starting the Client daemon

On UNIX and Linux, you can issue a command to start the Client daemon.

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.

# Chapter 61. Stopping CICS Transaction Gateway on UNIX and Linux

To shut down CICS Transaction Gateway, you must stop the Gateway and Client daemons separately.

When you stop CICS Transaction Gateway, stop the daemons in the following order:

1. The Gateway daemon. You can use either the **ctgd** or the **ctgadmin** command. For more information, see the "ctgd command reference" on page 364 and the "ctgadmin command reference" on page 361.

   For example:

   ```
   ctgadmin -a shut [-immediate] [-adminport <port>]
   ```

   If the default port is not used, you must use the **-adminport** option.
2. Client daemon. You stop the Client daemon with the **cicscli** command:

   ```
   cicscli -x
   ```

**Note:** Stopping the Client daemon while the Gateway daemon is still running is not supported. If the Client daemon is stopped 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.

## Stopping a Gateway daemon that is 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 362.

**Related information**:

Chapter 64, "Normal and immediate shutdown," on page 341
You can stop the IBM CICS Transaction Gateway normally, waiting for all work to finish, or immediately, without waiting.

# Chapter 62. Starting CICS Transaction Gateway on Windows

CICS Transaction Gateway is installed and runs as a Windows service, so it can be started and stopped in the same way as any Windows service.

The `IBMCICSTransactionGateway` service can be started manually from the command prompt, with the **ctgadmin** command, or automatically from the Windows services control panel. For more information about starting the service manually, see "ctgadmin command reference" on page 361 For more information about starting and stopping Windows services, refer to your Windows documentation. On Windows, the Gateway daemon and the Client daemon are started and stopped together.

The authority to start and stop the CICS Transaction Gateway service can be granted or denied to named users or groups by running the **ctgservice** command with the `-G` or `-D` options. For more information, refer to the "ctgservice command reference" on page 365.

The first time that you start the CICS Transaction Gateway service you start it manually. When you have successfully created and tested a configuration file, you should then configure the IBM CICS Transaction Gateway service to start automatically when Windows starts. The product does not require a user to be logged on to Windows for it to run. For more information, see "Configuring the Windows service" on page 95.

By default, CICS Transaction Gateway searches for the configuration file `ctg.ini`, in the `<product_data_path>`. If the configuration file is stored in a different location or has a different file name, use the *CICSCLI* system environment variable to define the fully qualified path and filename of the configuration file. For more information, see "Specifying the configuration file" on page 95.

If a local administration port is not explicitly configured, the Gateway daemon listens for administration requests on the default port. For more information, see "Port for local administration" on page 161. You can override the configured administration port by setting the **port** override parameter with the **ctgservice** command before you start CICS Transaction Gateway. For more information, see the "ctgservice command reference" on page 365.

## Setting CICS Transaction Gateway startup override options

On Windows, you can use the **ctgservice** command to register Gateway daemon startup override options and the **cicscli -r** command to register startup override options for the Client daemon.

Startup override options specified are checked when the IBM CICS Transaction Gateway service starts. The options are written to the Windows Application Event log.

### Gateway daemon startup options

Use the **ctgservice** command to specify options to override Gateway daemon parameter values specified in the configuration file. For more information, see the "ctgservice command reference" on page 365.

### Client daemon startup options

Use the **cicscli -r** command to specify Client daemon startup options. For more information, see "Setting startup override options" on page 355.

## Messages and error logs for CICS Transaction Gateway on Windows

How to find the messages and error logs for a CICS Transaction Gateway service running on Windows.

When starting CICS Transaction Gateway on Windows, any standard output or error text is captured in the file `jvmdbg.log`, in the `<product_data_path>` directory. The main use of this file is to capture debug output from the JVM, when the CICS Transaction Gateway is started with debug Java startup parameters specified.

Messages generated by CICS Transaction Gateway when running as a service are written to the application log and the system log of the Windows event viewer, whenever the service is started or stopped. Details of any startup override parameters are written to the Windows application event log when the service starts.

CICS Transaction Gateway can also be installed on a remote desktop server. In this situation you can access the product through remote desktop services.

# Chapter 63. Stopping CICS Transaction Gateway on Windows

You can stop the IBM CICS Transaction Gateway service from the Windows Services control panel or from a command prompt using the **ctgadmin** command.

Use the following command to stop the IBM CICS Transaction Gateway service from a command prompt:

```
ctgadmin -a shut [-immediate] [-adminport <port>]
```

**Related concepts**:

"ctgadmin command reference" on page 361
Options that can be used with the **ctgadmin** command. Options are not case sensitive.

**Related information**:

Chapter 64, "Normal and immediate shutdown," on page 341
You can stop the IBM CICS Transaction Gateway normally, waiting for all work to finish, or immediately, without waiting.

# Chapter 64. Normal and immediate shutdown

You can stop the IBM CICS Transaction Gateway normally, waiting for all work to finish, or immediately, without waiting.

## 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 shutting down.

## Immediate shutdown

When you force an immediate shutdown, CICS Transaction Gateway does not wait for any work to complete. This method of shutdown is not recommended, because it cannot be determined if work has completed and relies on connected systems to resolve in flight requests.

# Chapter 65. 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. Issue the command from a command prompt. See "ctgadmin command reference" on page 361 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: Developing Applications* for details of return codes used by the **ctgadmin** command.

## Controlling IPIC connections

Options are available to control the availability of IPIC connections.

These options allow a connection to be stopped, preventing further work from being sent to the CICS server. When the status is STOPPING or STOPPED, the connection can only be reconnected with a **START** command. New client requests will not trigger the Gateway daemon to reconnect.

To get help on these options, issue the following command:

```
ctgadmin -a server -?
```

*Table 28. Commands for controlling IPIC connections.*

| Option | Description |
|---|---|
| -start=<*IPIC Server*> | Start an IPIC server connection.<br><br>The Gateway daemon will attempt to connect to the CICS server and the status will become AVAILABLE or UNAVAILABLE. |

*Table 28. Commands for controlling IPIC connections. (continued)*

| Option | Description |
|---|---|
| `-stop=<IPIC Server>` | Stop an IPIC server connection.<br><br>During a normal stop, new work to the IPIC server is not allowed to start. A normal stop has two phases:<br>1. During the initiation phase, the Gateway daemon waits until all in-progress work on the connection has finished.<br>2. During the completion phase, the Gateway Daemon stops the IPIC connection.<br><br>The following requests are accepted on the IPIC connection during the initiation phase of a normal stop:<br>ECI requests (SYNC or ASYNC) which continue a logical unit of work.<br>ECI requests (SYNC or ASYNC) which commit or back out a logical unit of work.<br><br>All other requests are rejected.<br><br>Outstanding logical units of work prevent the connection from stopping. |
| `-stop=<IPIC Server> -immediate`<br>or<br>`-stop=<IPIC Server> -imm` | Immediately stop an IPIC server connection without waiting for work to complete. |

**Related reference**:

"CS*x*_CSTATUS values" on page 538
This table describes the possible values for CS*x*_CSTATUS.

## Controlling Gateway and JNI trace

Use the **ctgadmin -a trace** command to control Gateway daemon and JNI trace.

To control Gateway daemon trace use the following command:

```
ctgadmin -a trace [-tfile=<filename>][-tfilesize=<number>][-tlevel=<number>]
[-truncationsize=<number>][-dumpoffset=<number>]|-fulldatadump][-adminport=<port>]
```

Specifying -tlevel value between 1 and 4 turns Gateway trace on and specifying a value of 0 turns Gateway trace off.

**Note:** The **ctgadmin** Gateway daemon trace options have the restriction that once the trace is started only the trace level can be changed. To change the other options you must first switch the trace off by setting the trace level to 0.

To control JNI trace use the following command:

```
ctgadmin -a trace [-jnifile=<filename>][-jnilevel=<number>][-adminport=<port>]
```

Specifying -jnilevel value of 1 turns JNI trace on and specifying a value of 0 turns JNI trace off.

To query the current trace settings use the following command:

```
ctgadmin -a trace[-adminport=<port>]
```

If the default port is not used, you must use the -adminport option.

## 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 | Description |
|---|---|---|
| -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.<br><br>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.<br><br>On Windows if you specify a read-only network drive as the location of tfile, the CICS Transaction Gateway returns an "unknown" error, rather than indicating that the file is not writable. This is due to a limitation in the Java SDK. |
| -jnilevel | -jl | **0** Off. No trace information is output.<br><br>**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 `<product_data_path>/gateway.trc` on Windows platforms and `/var/cicscli/gateway.trc` on UNIX and Linux platforms. |

| Option | Short form | Description |
|---|---|---|
| -tlevel | -tl | Specifies the Gateway trace level. Permitted values are:<br>**0** Off. No trace information is output.<br>**1** Exception tracing. Only exceptions are traced. This is equivalent to the `-stack` option on the **ctgstart** and **ctgservice** commands. For more information, see Chapter 69, "Command reference," on page 361.<br>**2** Trace exceptions, and entry and exit of methods.<br>**3** Trace exceptions, some internals, and entry and exit of methods. This is equivalent to the `-trace` option on the **ctgstart** and **ctgservice** commands. For more information, see Chapter 69, "Command reference," on page 361.<br>**4** Full debug tracing (all trace points). This is equivalent to the `-x option` on the **ctgstart** and **ctgservice** commands. For more information, see Chapter 69, "Command reference," on page 361. |
| -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).<br>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.

| Option | Description |
|---|---|
| -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. |

## 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 | Description |
|---|---|---|
| -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 528
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 are 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 | Description |
|---|---|---|
| -disable | -dis | Disable all the request monitoring user exits. No exit will receive event notifications. |
| -enable | -ena | Enable all the request monitoring user exits. All active exits will receive event notifications |
| -command=<*command*> | -cmd=<*command*> | The command that will be sent to all configured and active request monitoring user exits. This is a string.<br><br>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 are 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 | Description |
|---|---|---|
| -command= *command* | -cmd= *command* | The command that is sent to the configured CICS request exit. This is a string.<br><br>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”. |

# Chapter 66. Client daemon administration

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 that are related to Client daemon administration.

The Windows command **cicscliw** performs the same functions as the **cicscli** command. However, it is a Windows program that displays its messages in a dialog box, rather than the command prompt.

On Windows, the Client daemon starts automatically when you start the Gateway daemon, and if you stop the Gateway daemon, the Client daemon also stops.

On UNIX and Linux, stopping the Gateway daemon does not stop the Client daemon; you must shut down the Client daemon after you stop the Gateway daemon. For more information, see Chapter 61, "Stopping CICS Transaction Gateway on UNIX and Linux," on page 335.

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 362.

## Controlling CICS server connections

TCP/IP and SNA connections to CICS servers, can be started or stopped using the **cicscli** command.

On Windows, you can start a connection to a CICS server after the IBM CICS Transaction Gateway service has been started. On UNIX and Linux, starting a connection to a CICS server will start the Client daemon if it is not already running. You do not have to start server connections explicitly. When a request is sent to a server, the server connection is automatically started, if it is not already established. If you change the configuration of a CICS server connection, you must stop and restart the connection.

To start a server connection to a server named *servername*, enter the following command:

```
cicscli -s=servername
```

To shut down the connection to a server named *servername*, allowing all outstanding units of work to first complete, enter the following command:

```
cicscli -x=servername
```

To shut down the connection to a server named *servername* immediately, without allowing all outstanding units of work to complete, enter the following command:

```
cicscli -i=servername
```

Shutting down a server connection does not shut down the Client daemon or connections to other servers.

## Shutting down the Client daemon

For UNIX and Linux only, 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 Client daemon for all connected servers, without completing outstanding units of work, enter:

```
cicscli -i
```

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 and can prevent normal operation of the Client daemon. If the Client daemon fails to shut down using the cicscli command, use the **kill -2** command.

## Restarting the Client daemon

On UNIX and Linux, you can shut down the Client daemon and then start it again.

If you want to shut down and restart the Client daemon, first shut down the Gateway daemon. Failure to do so causes both the Gateway daemon and the Client daemon to behave unpredictably. When the Gateway daemon has shut down, you can shut down and restart the Client daemon before restarting the Gateway daemon.

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.

To shut down the Client daemon for all connected servers immediately, without completing outstanding units of work, and then start it again, enter:

```
cicscli -j
```

The **cicscli -j** is equivalent to **cicscli -i** followed by **cicscli -s**. This does not reestablish server connections or trace settings.

## Starting client tracing

You can control Client daemon tracing using the **cicscli** command. You can improve performance while tracing by using memory mapped tracing.

### To Create a Binary Trace File

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 value of `size` must be in the range 0 - 32767. The default is 512 bytes.

On UNIX and Linux, you can start the Client daemon trace from startup, using either of these methods:

- Enter the trace parameters into the *CTGD_PARAMS* variable in the `ctgd.conf` file. For example, to set the trace to Protocol driver tracing level 2 enter:

  ```
  CTGD_PARAMS="-c-d -c-m=DRV.2"
  ```

  When you start the Client with the command `ctgd start`, trace starts automatically.
- Using the **ctgstart** command. For example, to set the trace to Protocol driver tracing level 2 enter:

  ```
  ctgstart -c-d -c-m=DRV.2
  ```

To trace the Client daemon from startup on Windows, set the trace options as startup parameters. For more information, see "Setting CICS Transaction Gateway startup override options" on page 337.

The Client daemon writes trace entries to a file, the default file name is `cicscli.bin`. On UNIX and Linux systems the trace file is created in the `/var/cicscli` directory. On Windows systems the trace file is created the `<product_data_path>` directory.

### Memory Mapped 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 501. For important security information, see "Security considerations for UNIX and Linux systems" on page 353.

To use memory mapped tracing, do the following:

1. Turn on wrapping trace by setting the "Client trace file wrap size" on page 235 in the configuration file to a value greater than 0.
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 502.

## 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.

To display the list of components set enter the command:

```
cicscli -m
```

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 502. |
| 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=<components> -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 Chapter 47, "Configuring trace settings," on page 233.

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
```

On Windows, trace stops automatically if you shut down the IBM CICS Transaction Gateway service. On UNIX and Linux, trace stops automatically if you shut down the Client daemon.

## Security considerations for UNIX and Linux systems

The Client daemon defines the access permissions to the client trace and log files. These files are created in the `/var/cicscli` directory.

### Client daemon administration

To restrict Client daemon administration to the root user modify the permissions on the `<install_path>/bin/cicscli` executable. Use a command such as:

```
chmod 700 /opt/ibm/cicstg/bin/cicscli
```

### Log and trace files

The `/var/cicscli` directory is created by the installer with permissions 777 this allows all users write access to the `/var/cicscli` directory. A user with write access to the `/var/cicscli` directory can delete files from that directory regardless of the permissions on the files. If you do not want users to have the ability to delete trace and log files remove their write access to the `/var/cicscli` directory. For example, a command such as:

```
chmod 755 /var/cicscli
```

allows users to see files in `/var/cicscli` directory but not to create, delete, or move them. After restricting access to `/var/cicscli`, users will only be able to start the Client daemon if the Client daemon log and trace files already exist.

To stop users having read access to the files in `/var/cicscli` use a command such as:

```
chmod 711 /var/cicscli
```

The Client daemon defines the access permissions to the client trace file, permissions vary with the type of trace being processed. When processing memory mapped trace the Client daemon defines trace file permissions as 666, all users have read and write access. Memory mapped trace is started with the `-b` option. (See “Starting client tracing” on page 350.) When processing basic trace the Client daemon defines trace file permissions as 622, all users have write access and only the owner can read it for formatting.

The Client daemon prevents you from starting tracing if an unauthorized user has deleted and recreated the Client daemon trace file.

## 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.

## Setting startup override options

On Windows, you can set startup override options for the Client daemon.

Set Client daemon startup override options as follows:

```
cicscli -r [-A<parameter1> [-A<parameter2>]]
```

**Example 1**: the following command initiates a server connection using a specific user name and password:

```
cicscli -r -A-s=servername -A-u=username -A-p=password
```

To clear any saved startup parameters use the **cicscli -r** command with no parameters.

**Example 2**: the following command initiates memory mapped trace for all Client daemon trace components:

```
cicscli -r -A-b -A-d -A-m=all
```

**Note:** memory mapped trace requires the Client trace file wrap size to be defined. For more information, see "Client trace file wrap size" on page 235.

For information on the **cicscli** parameters that can be set using the **cicscli -r**, see "cicscli command reference" on page 362.

**Related information**:

"Setting CICS Transaction Gateway startup override options" on page 337
On Windows, you can use the **ctgservice** command to register Gateway daemon startup override options and the **cicscli -r** command to register startup override options for the Client daemon.

## Viewing startup override options

On Windows, you can view startup override options for the Client daemon.

To view startup override options for the Client daemon, enter:

```
cicscli -t
```

## 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.

# Chapter 67. Understanding 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 timeouts 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

**Note:** Absolute times provided by the statistics APIs and request monitoring exits, and timestamps written to logs and traces, are obtained from the operating system.

# Chapter 68. Viewing message help

Help is available for CICS Transaction Gateway messages by using an option on the **ctgadmin** command.

To obtain help for a message, enter the `ctgadmin` command followed by the `-msg` option, then the message number:

```
ctgadmin -msg message ID
```

Where *message ID* has the format [CTG|CCL]*nnnn*[I|E|W], where:

- The message prefix CTG or CCL is optional
- The message number *nnnn* comprises 1 through 4 decimal digits and is mandatory
- The message suffix I, E or W is optional

To obtain help for the `-msg` option itself:

```
ctgadmin -msg -?
```

Help for error and warning messages is detailed and includes the following information:

- message number
- message text
- explanation
- system action
- user response

Help for information messages includes the following information:

- message number
- additional information

**Related information**:

Messages

# Chapter 69. Command reference

This topic lists the commands needed to operate CICS Transaction Gateway.

## 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>]
```

Issue the following command for general help:

```
ctgadmin -?
```

To get help on a particular action, issue a command like the following:

```
ctgadmin -a action -?
```

The options are:

| Option | Description |
|---|---|
| -a dump | "Dumping diagnostic information" on page 346. |
| -a trace | "Controlling Gateway and JNI trace" on page 344. |
| -a stats | "Querying statistics" on page 347. |
| -a crexit | "CICS request exit control" on page 348. |
| -a rmexit | "Request monitoring exit control" on page 348. |
| -a server | "Controlling IPIC connections" on page 343. |
| -a start | Chapter 62, "Starting CICS Transaction Gateway on Windows," on page 337. |
| -a shut | Chapter 63, "Stopping CICS Transaction Gateway on Windows," on page 339.<br><br>Chapter 61, "Stopping CICS Transaction Gateway on UNIX and Linux," on page 335. |
| -msg | Chapter 68, "Viewing message help," on page 359. |
| -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. |

### Return codes from the ctgadmin command

The **ctgadmin** command can be run through a script. The return codes listed in this information are for errors that can be dealt with by the user. Return codes not in the following list indicate an internal processing error, and are used by the service organization in the event of a persistent problem.

| Return code | Description |
|---|---|
| 0 | The command completed successfully, or help was requested |
| 1 | The product is not correctly installed |
| 2 | Failure reading Windows registry key |
| 3 | Failure writing Windows registry key |
| 4 | Java not found on system |
| 5 | User is not authorized to start the CICS TG service |
| 11 | Java Version not supported |
| 13 | The product is not correctly installed; `ctgadmin.jar` cannot be located |
| 14 | Operating system not supported |
| 100 | Command failed due to bad parameter specification |
| 101 | Failure communicating with the CICS Transaction Gateway |
| 102 | Attempt to connect on non admin port |
| 104 | Failure to locate messages file |
| 150 | Authorization failure |

## `ctgassist` command reference

You can use the **`ctgassist`** command to generate a web service binding file and one or more language data structures that you can use with web service applications.

The **`ctgassist`** utility installed as part of the product uses the installed Java. The **`ctgassist`** utility installed as part of the SDK requires Java 7 or later to be available on the path.

The format of the **`ctgassist`** command is:

```
ctgassist [-X] <parameter file>
```

where:

- `-X` is an optional switch to turn on tracing in the assistant
- `<parameter file>` is the name of a file that contains the parameters in the form of `<name>=<value>` pairs.

**Note:** You must provide the name of the parameter file. If you do not provide a parameter file, the command fails with a message indicating that a mandatory parameter is missing.

## cicscli command reference

Options that can be used with the **`cicscli`** command.

On UNIX and Linux the syntax of the **`cicscli`** command is:

```
cicscli [[-s=servername]|[-x=servername]|[-i=servername]]
[-l]
[[-d[=nnn] [-b] [-m[=components]]]|-o]
[-c=servername [-u[=userid]] [-p[=password]]]
[-w|-q]
[-v]
[-y|-j]
```

On Windows the syntax of the **cicscli** command is:

```
cicscli [[-s=servername]|[-x=servername]|[-i=servername]]
[-l]
[[-d[=nnn] [-b] [-m[=components]]]|-o]
[-c=servername [-u[=userid]] [-p[=password]]]
[-r [-a<parameter1> [-a<parameter2>...]][-t]
[-w|-q]
[-v]
```

All options of the form *-x=variable* can contain spaces in the variable part, if the variable part is enclosed in double quotation marks. Double quotation marks within variables must be entered as \" , that is with a backslash preceding the double quotation mark.

Enter **cicscli -?** to display help for the **cicscli** command.

The options are:

| Option | Description | Option valid with -r parameter |
|---|---|---|
| -b | "Starting client tracing" on page 350. | Yes |
| -c=servername | "Setting security for server connections" on page 354. | Yes |
| -d[=nnn] | "Starting client tracing" on page 350. | Yes |
| *-i=<servername>* | "Controlling CICS server connections" on page 349. | No |
| *-j* | UNIX and Linux only.<br><br>Restart the Client daemon after an immediate shutdown. | No |
| -l | "Listing the connected servers" on page 355. | No |
| -m[=components] | "Specifying the trace components" on page 351. | Yes |
| -o | "Stopping client tracing" on page 353. | No |
| -p=password | "Setting security for server connections" on page 354. | Yes |
| -q | "Controlling cicscli command messages" on page 354. | No |
| -r [a<parameter1> [-a<parameter2>...]] | Windows only."Setting startup override options" on page 355. | No |
| *-s=<servername>* | "Controlling CICS server connections" on page 349. | Yes |
| -t | "Viewing startup override options" on page 355. | No |

| Option | Description | Option valid with -r parameter |
|---|---|---|
| -u=userid | "Setting security for server connections" on page 354. | Yes |
| -v | "Displaying the version of CICS Transaction Gateway" on page 354. | No |
| -w | "Controlling cicscli command messages" on page 354. | No |
| *-x=<servername>* | "Controlling CICS server connections" on page 349. | No |
| *-y* | UNIX and Linux only.<br><br>Restart the Client daemon after a normal shutdown. | No |

## ctgd command reference

You can use the **ctgd** command on on UNIX and Linux to start CICS Transaction Gateway as a background task.

The command **ctgd start** creates log file `/var/cicscli/ctgd.log`.

The syntax is:

```
ctgd {start|stop}
```

The **ctgd** command uses a configuration file to define environment variables that are used when starting the Gateway daemon and Client daemon processes. The default location and name for the **ctgd** configuration file is `/var/cicscli/ctgd.conf`. Use the environment variable `CTGDCONF` to specify an alternative configuration file for **ctgd**. If the `ctgd.conf` configuration file does not exist, the **ctgd** command will use default values.

*Table 29. ctgd configuration file variables*

| Parameter | Description |
|---|---|
| CTGD_USER | Run the CICS TG as user `$CTGD_USER`. If you do not define this variable, CICS TG runs under the user who started **ctgd**. At system startup this is root. An error will be generated if the **ctgd** command cannot change to this user. |
| CTGD_GROUP | Run the CICS TG as group `$CTGD_GROUP`. If you do not define this variable, CICS TG runs under the primary group of the user who started **ctgd**. An error will be generated if the **ctgd** command cannot change this group. |

*Table 29. ctgd configuration file variables (continued)*

| Parameter | Description |
|---|---|
| CTGD_ADMINPORT | TCP port which listens for local administration requests from **ctgadmin**. If not specified, the **adminport** parameter specified in the configuration file `ctg.ini` is used. If the **adminport** parameter is not defined in the configuration file, the default value is 2810. |
| CTGD_PARAMS | Specify Gateway daemon startup override options. For more information about the **ctgstart** options, see "ctgstart command reference" on page 369. Any **ctgstart** option can be specified, except the **-adminport** override. To specify a custom administration port, use the `CTGD_ADMINPORT` variable. When specifying multiple options, separate them with a space and enclose the entire CTGD_PARAMS value in quotation marks. A sample **ctgd** configuration file is installed as `<install_path>/samples/configuration/ctgdsamp.conf`. |

You can also set any environment variables in the `ctgd.conf` file. For more information, see Chapter 49, "Environment variables reference," on page 249.

## ctgservice command reference

You can use the **ctgservice** command on Windows to set Gateway daemon override parameters and to grant or deny start and stop authority on the IBM CICS Transaction Gateway service.

Setting Gateway daemon override parameters or defining start and stop authority requires that the **ctgservice** command is run by an administrator. Using **ctgservice** from a command prompt does not start the IBM CICS Transaction Gateway service.

The IBM CICS Transaction Gateway service is created with a default security descriptor. The access rights of the default security descriptor are defined by the Windows service control manager, for more information refer to your Windows documentation. You can use the **ctgservice** command to modify the default access rights to give users or groups the authority to start and stop the IBM CICS Transaction Gateway service.

The **ctgservice** command sets the Gateway daemon override parameters to be used when the IBM CICS Transaction Gateway service is started. These override parameters override the Gateway daemon parameter values specified in the configuration file. The override parameters are validated when the service is started and not when they are set by the **ctgservice** command. The **ctgservice** command uses the JVM specified by the **ctgjava** command. For more information, refer to "Specifying the JVM" on page 95.

The syntax of the **ctgservice** command is:

```
ctgservice -?|-T|-G Name|-D Name|-R [-ACTGParam1 [-ACTGParam2] [-A-jArgument]...]
```

where:

**-?** Displays the command syntax and the list of options that can be used with **ctgservice**.

**-T** Displays the registered parameter overrides and the users and groups that can start and stop the service.

**-G <Name>**
Grants a user or group the authority to start and stop the IBM CICS Transaction Gateway service, by allowing access rights: SERVICE_QUERY_CONFIG, SERVICE_QUERY_STATUS, SERVICE_START and SERVICE_STOP. If the name includes spaces, then it must be enclosed in quotes.

**-D <Name>**
Denies a user or group the authority to start and stop the IBM CICS Transaction Gateway service, by denying access rights: SERVICE_START and SERVICE_STOP. If the name includes spaces, then it must be enclosed in quotes.

**-R** Registers the override parameters to be used when starting the Gateway daemon. The `-R` option resets the registered override parameters. If you use **ctgservice -R** with no following options, all override parameters are removed. To define override parameters for the Gateway daemon the `-A<override>` option must be used with the `-R` option.

**-A<override>**
Specifies a startup override parameter for the Gateway daemon or an argument for the Gateway daemon JVM. The `-A<override>` option must be used with the `-R` option.

**Note:** The **-G** and **-D** parameters can only be specified by a user with administrator authority. These parameters update the Access Control List (ACL) for the CICS TG service. A user or group is granted or denied permission to start and stop the CICS TG service. The **-T** parameter has been updated to list the access control list entries that are associated with the CICS TG service.

For example:

- To grant "Power Users" the authority to start and stop the IBM CICS Transaction Gateway service:

  ```
  ctgservice -G "Power Users"
  ```

- To deny the "Guests" group from starting and stopping the IBM CICS Transaction Gateway service.

  ```
  ctgservice -D Guests
  ```

- To set override parameters for the Gateway daemon.

  ```
  ctgservice -R -A-x -A-tfile=C:\temp\ctgtrace.txt -A-j-Dgateway.T.setJNITFile=
  C:\temp\ctgJNItrace.txt
  ```

The following Gateway daemon overrides are available to be set using the **-R** option

| Override parameter | Description |
|---|---|
| -port=*number* | Specifies the TCP/IP port number on which the Gateway daemon will listen. |
| -httpport=*port number* | Specifies the TCP/IP port on which the Gateway daemon listens for HTTP requests. |

| Override parameter | Description |
| --- | --- |
| -httpsport=*port number* | Specifies the TCP/IP port on which the Gateway daemon listens for HTTPS requests. |
| -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 on Unix, or the **ctgservice**- command line. |
| -adminport=*number* | Specifies the port used to communicate with the Gateway daemon when controlling the Gateway daemon through the **ctgadmin** command on Unix or the **ctgservice** command on Windows. |
| -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 Chapter 81, "Tuning the Gateway," on page 415 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 Chapter 81, "Tuning the Gateway," on page 415 for performance information. |
| -initworker=*number* | Specifies an initial number of worker threads. See Chapter 81, "Tuning the Gateway," on page 415 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 Chapter 81, "Tuning the Gateway," on page 415 for performance information. |

| Override parameter | Description |
|---|---|
| -trace | Enables standard tracing (see Chapter 102, "Tracing," on page 499). 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 top of the trace output.<br><br>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 host names" on page 168 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 Chapter 102, "Tracing," on page 499). 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.<br><br>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. |

| Override parameter | Description |
|---|---|
| -stack | Enables Java exception stack tracing (see Chapter 102, "Tracing," on page 499). Java exceptions are traced, including those expected during typical operation. Expected exceptions include:<br>• An IOException resulting from a Java Client application ending or disconnecting from a network protocol.<br>• An InterruptedException resulting from a CICS Transaction Gateway protocol handler timeout such as the idle timeout, or a ping timeout.<br>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. On Windows, this option replaces the `-X` option used in earlier versions of the CICS Transaction Gateway. You can pass multiple arguments to the JVM. Specify the `-j` option multiple times to pass multiple arguments to the JVM. |
| -classpath=*class path* | Specifies additional entries to append to JVM class path that are used when launching the JVM. For example, the location of a jar file containing request exits. |
| -requestExits=*exits* | List of classes to be used for request exit monitoring. |

# ctgstart command reference

You can use the **ctgstart** command on UNIX and Linux to start CICS Transaction Gateway with options.

The format of the **ctgstart** command is:

```
ctgstart <options> [<-j>JVMArg1 <-j>JVMArg2...] [<-c>CicscliArg1 <-c>CicscliArg2...]
```

To list the startup options and their purposes, type **ctgstart -?**

If you are running the Gateway daemon as a background process, these startup options, except for `-adminport`, can be specified on the **ctgd** parameter **CTGD_PARAMS**. For more information, see "ctgd command reference" on page 364.

## Startup options

| Option name and syntax | Description |
|---|---|
| *-port=number* | Specifies the TCP/IP port number on which the Gateway daemon will listen. |

| Option name and syntax | Description |
|---|---|
| *-sslport=number* | Specifies the TCP/IP port number on which the Gateway daemon will listen for SSL requests. |
| *-httpport=port_number* | Specifies the TCP/IP port on which the Gateway daemon listens for HTTP requests. |
| *-httpsport=port_number* | Specifies the TCP/IP port on which the Gateway daemon listens for HTTPS 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. |
| *-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 Chapter 81, “Tuning the Gateway,” on page 415 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 Chapter 81, “Tuning the Gateway,” on page 415 for performance information. |
| *-maxhttpconnect=number* | Specifies the maximum number of HTTP and HTTPS client threads |
| *-initworker=number* | Specifies an initial number of worker threads. See Chapter 81, “Tuning the Gateway,” on page 415 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 Chapter 81, “Tuning the Gateway,” on page 415 for performance information. |

| Option name and syntax | Description |
|---|---|
| *-trace* | Enables standard tracing (see Chapter 102, "Tracing," on page 499). 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.<br><br>Trace output is written to `/var/cicscli/gateway.trc`, unless you use the `-tfile` option. 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.<br><br>See Note. |
| *-quiet* | Disables the reading of input from the console and disables writing to stdout. When the -quiet option is used, the Gateway daemon log destination must be set to `file`. For more information, see "Gateway daemon logging" on page 161 |
| *-dnsnames* | Enables the display of symbolic TCP/IP host names in messages. See "Display TCP/IP host names" on page 168 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 Chapter 102, "Tracing," on page 499). 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.<br><br>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. |

| Option name and syntax | Description |
|---|---|
| *-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 Chapter 102, "Tracing," on page 499). Java exceptions are traced, including those expected during typical operation. Expected exceptions include:<br>• An IOException resulting from a Java client application ending or disconnecting from a network protocol.<br>• An InterruptedException resulting from a CICS Transaction Gateway protocol handler timeout such as the idle timeout, or a ping timeout.<br>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=myserver -c-d -c-m=all` will call `cicscli -s=myserver -d -m=all`.<br>The server name must be supplied.For details about which options can be passed to the Client daemon, see Chapter 66, "Client daemon administration," on page 349 section. |
| *-requestExits=exits* | A comma separated list of one or more fully qualified class names of the request monitoring exits. |

**Note:** The trace options specified on the **ctgstart** command must specify the full set of options required for trace. If you specify one trace option the default values for the other options will be used and all trace options in ctg.ini will be ignored.

# Part 9. 3270 terminal emulation and printing

CICS Transaction Gateway provides cicsterm, a CICS 3270 terminal emulator and cicsprnt, a CICS 3270 printer terminal emulator.

# Chapter 70. cicsterm emulator

cicsterm is a CICS 3270 terminal emulator that can be used for running CICS applications without the need for a separate 3270 emulator product.

## Using cicsterm

The cicsterm command enables you to work with terminal emulator sessions.

The screen color attributes and keyboard settings of the terminal emulator can be customized with mapping files. This is useful for example if keyboard layouts are required to comply with a company standard. Terminal definitions can be automatically installed on the CICS server and do not have to be predefined.

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 227.
- 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 229.
- Run multiple terminal emulation sessions to one or more CICS servers at the same time.

You cannot connect to CICS over IPIC using the cicsterm command because IPIC does not support 3270 data flows. See Chapter 13, “Which protocol can be used?,” on page 39 for details of which protocols you can use to connect to CICS when using cicsterm.

### Windows platforms

When you run cicsterm the system detects whether the hardware on which the Client daemon is running is enabled for double-byte character set (DBCS) display. If DBCS is enabled, the terminal emulator can display DBCS characters. Some servers and protocols require a model terminal definition for the terminal emulator. The model terminal definition specifies explicitly that cicsterm must display DBCS characters.

If you are using a mouse, a single click moves the cursor to the position of the mouse pointer. A double click when a transaction is in progress is equivalent to pressing the Enter key. A double click when a transaction is not in progress has no effect.

## 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: Developing Applications*.

**-c=***colorfile*
Specifies a color mapping file to be used with the emulator; this overrides the default color mapping file `<install_path>/bin/cicscol.ini`, and any color mapping file specified by the **CICSCOL** environment variable. See "Customizing the screen colors for cicsterm" on page 229 for more details.

**-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 229 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 quotation marks ("). Any double quotation marks within the name of the file must be entered as backslash double quotation mark (\").

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*
Specifies a keyboard mapping file to be used with the emulator; this overrides the default key mapping file `<install_path>/bin/cicskey.ini` and any key mapping file specified by the **CICSKEY** environment variable. For more information, see "Keyboard mapping for cicsterm" on page 227

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 IBM AIX systems use `aixterm` in preference to `xterm`.

**-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.

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 quotation marks (“). Any double quotation marks within the command must be entered as backslash double quotation mark (\“).

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.

## 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: Developing Applications*.

## 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: Developing Applications* for more information.

## **cicsterm** restrictions

There are some restrictions when you use the **cicsterm** command.

### UNIX and Linux

- 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.

### Windows

- The OUTLINE extended attribute is supported only on DBCS terminals.
- 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.

# Chapter 71. 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]
[-?]
```

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.

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 |
| -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. On Windows this is equivalent to using the CICS Terminal Start menu shortcut.

# Chapter 72. cicsterm preferences on Windows

The terminal emulator provides menu options that can be used to customize the font, print the screen, and exit with save settings.

If you share a computer with others, or are a user of multiple computers on a network, your preferences are used whenever and wherever you launch terminal emulator. This functionality is provided by means of Windows roaming user support.

The CICSTERM.INI file is not erased when you uninstall CICS Transaction Gateway. Therefore, when you re-install CICS Transaction Gateway, the emulator uses the settings from the old CICSTERM.INI files.

The terminal emulator has a menu bar with **File** and **Settings** menus.

The commands on the **File** menu are:

**Print** Prints the emulator screen. This has the same effect as pressing the Print Screen key.

**Exit** Stops the terminal emulator.

The commands on the **Settings** menu are:

**Font** Opens a standard Windows dialog box that allows you to select any of the fixed-pitch fonts installed on your system. Select a font, a font style, and a size, and then select **OK**. The terminal window is then resized according to your font selection.

**Autosize**
Specifies that the size of TrueType fonts is adjusted according to the size of the terminal window. That is, when you maximize, minimize, or drag the borders to resize the window, the font is resized to match the new window size. This command has no effect if the current font is not a TrueType font.

**Save on Exit**
Specifies that the current font, terminal window size, and terminal position are saved, for use in future sessions, when you close the terminal window. If the **Settings > Save on exit** menu option is selected, your choices are stored in CICSTERM.INI, for use in future sessions. A copy of CICSTERM.INI is created for each Windows user account as follows:

*APPDATA*\IBM\CTG\CICSTERM.INI

where APPDATA is the environment variable defined for each user by the Windows operating system.

# Chapter 73. `cicsprnt` emulator

**cicsprnt** is a CICS 3270 printer terminal emulator that can be used by CICS server applications to send output to an attached printer.

## Using cicsprnt

Output can be sent to a printer attached to a port, such as LPT1. Alternatively, a command can be issued to process the data into a format more suitable for specialized printers.

You can have multiple printer terminal emulators running simultaneously.

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.

The maximum screen size is 27 rows by 132 columns. This is because the Client daemon supports the ASCII-7 subset of the 3270 data stream architecture, which supports only 12 bit addressing.

## `cicsprnt` options

A number of options and parameters are available when using the **cicsprnt** command to start a 3270 printer terminal emulator.

The syntax of the command is:

```
cicsprnt -m=modelname|-n=netname
[-s=servername|-r[=servername]]
[-t=[initialtransid]]
[-p=printcmd|-f=printfile]
[-q|-w]
[-j]
[-z]
[-b]
[-?]
```

The options are:

**-?** Causes the parameter syntax to be listed; any other options specified are ignored.

**-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 quotation marks (“). Any double quotation marks within the name of the file must be entered as backslash double quotation mark (\“).

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. On Windows enter the full path to the command, even if it is in a directory that appears in the search path.

If the command contains embedded blanks, the command must be surrounded by double quotation marks (“). Any double quotation marks within the command must be entered as backslash double quotation mark (\“).

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.

**-r=***servername* **or -s=***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.

## 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: Developing Applications*.

## 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 does not support the following commands:
  - `EXEC CICS RETURN TRANSID(name) IMMEDIATE`
  - `EXEC CICS RETURN TRANSID(name) IMMEDIATE INPUTMESSAGE(data-area)`
- cicsprnt on UNIX and Linux platforms does not support being interrupted using the `SIGHUP` signal or being controlled using the `SIGTSTP` and `SIGCONT` signals. Use of these signals might result in unpredictable behaviour. To start cicsprnt as a background task, cicsprnt must be started using the **nohup** utility. For example:

  ```
  nohup cicsprnt -s=<servername> -m=<model> -j -z > /var/cicscli/cicsprnt.nohup.out
  ```

# Chapter 74. 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 following table shows the tasks and associated parameters that the **cicsprnt** command supports:

| Option | Description |
|---|---|
| -? | Display help. |
| -b | Causes blank lines in the data stream to be printed. |
| -f | Specify a file to which print files are appended. The default location is <install_path>\bin on Windows and <install_path>/bin on UNIX and Linux. |
| -j | Issue one print job per transaction. |
| -m and -n | Define the 3270 printer terminal emulator characteristics. |
| -p | Determine the print file processing. |
| -q | Inhibit all output messages. |
| -r and -s | Start a 3270 print terminal emulator. |
| -t | Specify the initial transaction. |
| -w | Wait for confirmation before completing. |
| -z | Suppress the insertion of a blank character between single-byte and double-byte characters. |

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` on Windows and `<install_path>/bin` on UNIX and Linux).

**-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** options" on page 383.

On Windows the cicsprnt process runs as a minimized window, which can be enlarged to view the printer status. Use the pull-down menu to terminate the print function.

# Part 10. Security

Security mechanisms include link, bind and user security on connections, SSL client authentication, SSL server authentication, and identity propagation.

# Chapter 75. Security considerations

CICS Transaction Gateway can perform authentication and authorization checks at different points during the processing of requests.

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, IBM WebSphere Application Server and CICS Transaction Gateway are both running on Windows. The EJB application in IBM WebSphere uses the ECI resource adapter and the Client daemon to access the CICS COMMAREA application.

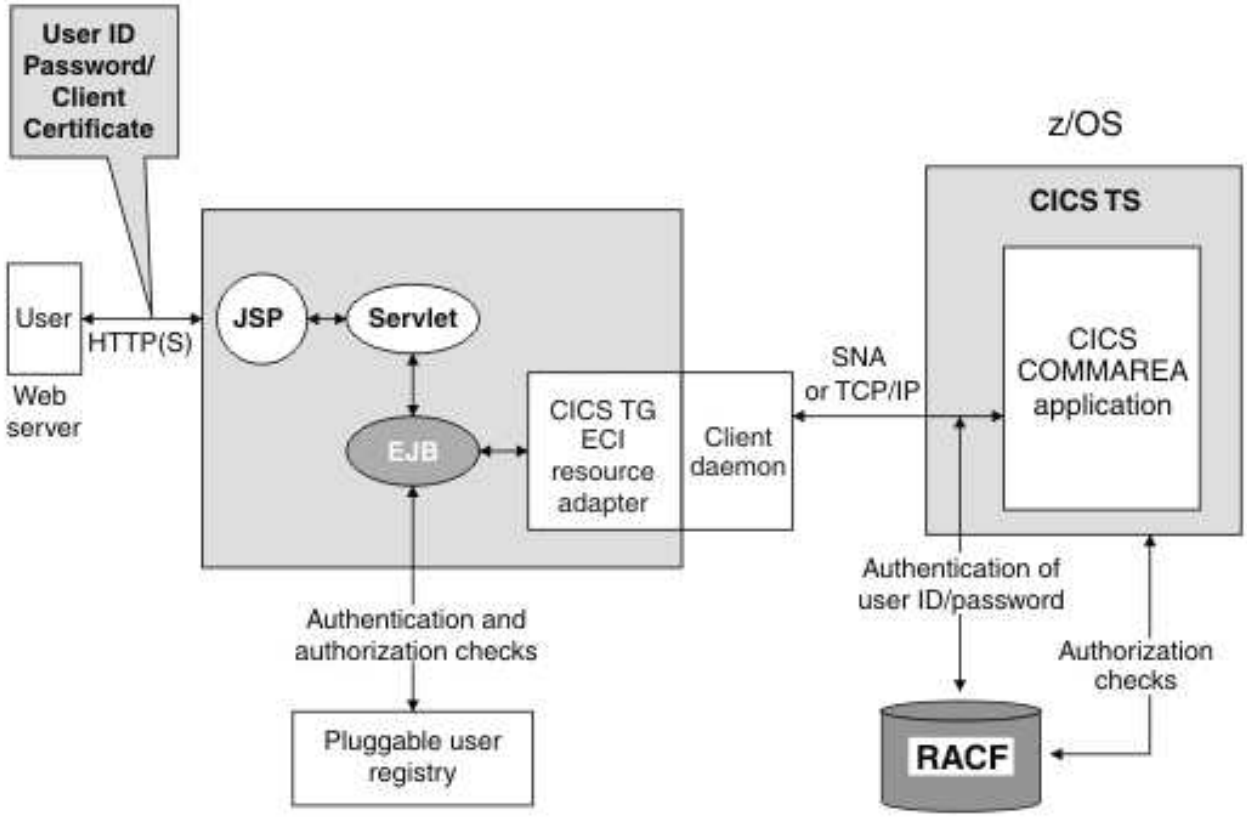



*Figure 27. Security in a local mode topology*

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 IBM WebSphere Application Server. The level of data encryption, server authentication and client authentication can be specified.

# Chapter 76. 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.

### 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 in the IPCONN definition in CICS. You can use an SSL certificate if you have a client authenticated SSL connection. 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 documentation.

### Bind security

For IPIC connections bind security is implemented using a client authenticated SSL connection. In this configuration the Java client application or CICS Transaction Gateway need to be authenticated by the CICS server before they are able to successfully connect. This prevents an unauthorized 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 ID 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 ID 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 Chapter 79, “Identity propagation,” on page 407.

## 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 prevents a remote user from attaching a transaction or accessing a resource for which the link user ID has no authority. When link security is in use, each session is given an authority defined by a link user ID. 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.

To specify link security, set a link user ID in the SECURITYNAME parameter of the CONNECTION definition on the CICS server, or specify a USERID parameter in the SESSIONS definition on the CICS server.

If a failure occurs in establishing link security, the link is given the security of the CICS server's default user.

### Bind security

Bind security prevents an unauthorized remote system from connecting to CICS. For SNA, this is termed session security and a check is made when there is a request to establish an SNA session with a remote system; that is, when the session is bound. For SNA sessions to CICS, bind security is controlled in IBM VTAM and in Communications Server using SERVAUTH profiles.

### User security

User security requires that incoming transaction attach requests supply a user ID and password. User security can never increase a user's authority above that of the link. Specify the ATTACHSEC=VERIFY on the CONNECTION definition on the

CICS server. If user security is not required set ATTACHSEC=LOCAL. SNA connections do not support the use of ATTACHSEC=IDENTIFY.

## TCP/IP connection security

TCP/IP connections to CICS servers can be secured with 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 is not supported for TCP/IP CICS server connections.

### Bind security

Bind security prevents unauthorized remote systems from connecting to CICS. For TCP/IP CICS server connections, bind security is controlled using firewall rules.

### User security

User security ensures that user IDs and passwords are supplied for all incoming transaction attach requests to CICS. To specify user security, set **ATTACHSEC=VERIFY** on the TCPIPSERVICE definition on the CICS server. If user security is not required set **ATTACHSEC=LOCAL**. TCP/IP connections do not support the use of **ATTACHSEC=IDENTIFY**.

## Default connection security

A default user ID and password can be set for each TCP/IP and SNA server connection.

By default, if no security credentials are supplied, transactions are run using the default user ID for the CICS server, if a default user ID is defined. For CICS Transaction Server for IBM z/OS, the default user ID is used only 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 CICS server and a user ID and password must be supplied. In all cases, transactions execute in the server with the authorities assigned to the user ID authenticated.

Use the **cicscli** command to set default security for TCP/IP and SNA server connections. Default security cannot be set for IPIC connections.

The default user ID and password can be set using one of the following methods:

- **cicscli** commands:

  ```
  cicscli -c=servername -u=userid -p=password
  ```

- From C applications, use the ESI function **CICS_SetDefaultSecurity**. This call is not available from the Java APIs.
- From C++ applications, use the **makeSecurityDefault** method of the **CclConn** or **CclTerminal** class.
- From COM applications on Windows, use the **MakeSecurityDefault** method of the **Connect** or **Terminal** COM class.
- On Windows platforms, through a Client daemon security window, see "Security pop-up windows" on page 396.

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. The default user ID and password cannot be set from the Java APIs.

**Note:** If the Client daemon is running in an environment where it survives user logoff, for example, as a Windows service, 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.

## Security pop-up windows

On Windows platforms CICS Transaction Gateway can be configured to display security pop-up windows for TCP/IP and SNA connections.

Security pop-up windows are displayed to prompt for connection security credentials if the following conditions are true.

- The Windows system is not a Remote Desktop Server.
- The Enable pop-up windows field in the Configuration Tool is selected; see "Enable pop-up windows" on page 183.
- The IBM CICS Transaction Gateway service is enabled to interact with the desktop.
- The server requires that a user ID and password are flowed in transaction attach requests.
- No valid default security settings are defined.
- The transaction being invoked is CCIN (client installation) or CTIN (terminal installation).
- A user ID and password have not been supplied on a terminal installation request.

Values entered via a security pop-up window, once verified, are used to set the default user ID and password values for that server connection.

Security pop-up windows are not displayed directly by ECI or EPI transaction requests, although they can be prompted by these requests, if the request causes a server connection to be established, or a terminal to be installed.

# Chapter 77. Connection security and SSL

CICS Transaction Gateway can secure a network connection using SSL (Secure Sockets Layer).

Java clients can connect over SSL to the Gateway daemon. In addition, SSL can be used in both local mode and remote mode for IPIC connections to CICS.

## 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:

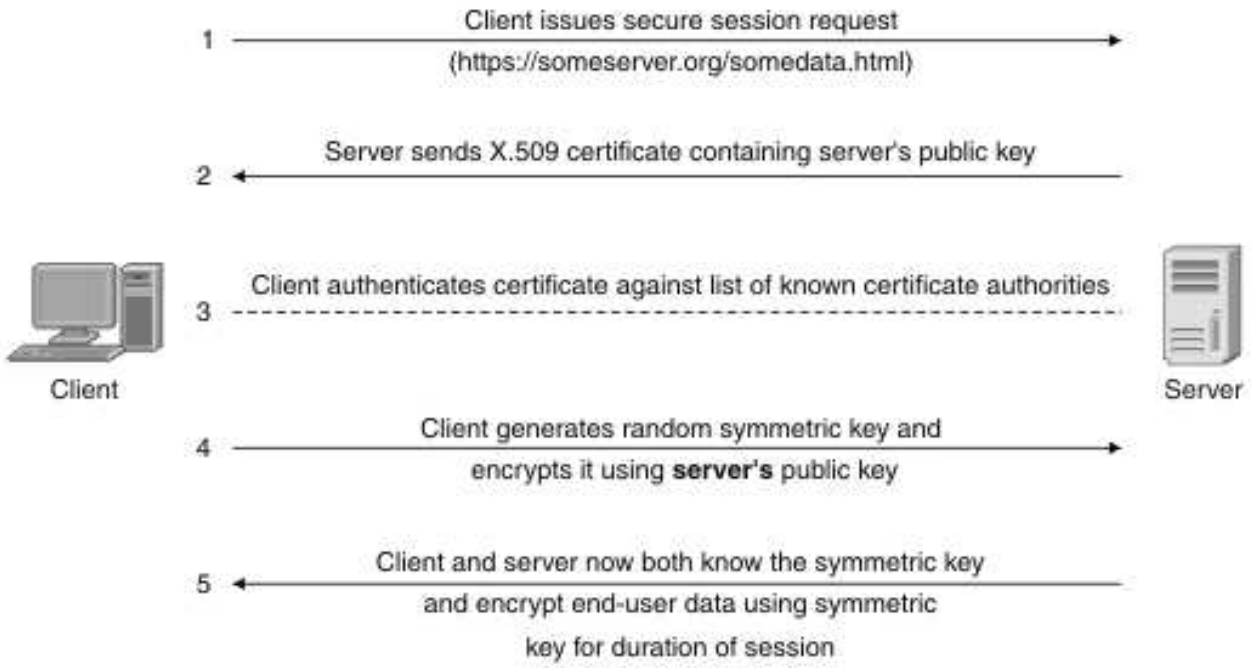



*Figure 28. SSL handshake*

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 Part 10, “Security,” on page 389.

## 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 documentation 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 Chapter 41, “Configuring SSL,” on page 197.

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 Chapter 41, "Configuring SSL," on page 197.

## 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 System Initialization Parameter, ENCRYPTION. For more information, see *CICS Transaction Server, System Initialization Parameters*.

# Chapter 78. 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** parameters 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.
- COM programs (Windows Only):

  Use the **AlterSecurity** method of the **Terminal** COM class. This can only be used for sign-on incapable created terminals.
- 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**. On Windows, you are prompted specifically for any unknown user ID, incorrect password, or expired password, after you enter your user ID and password in the security dialog.

The ESI functions can be used only with CICS servers that support password expiry management (PEM). See Chapter 10, “Supported software,” on page 27 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.
- COM programs (Windows Only):

  Use the **SetTermDefns** method of the **Terminal** COM class.
- 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 IBM z/OS

CICS Transaction Server for IBM 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.

## IBM TXSeries servers

IBM TXSeries servers support only sign-on capable terminals. A terminal installation request always results in a sign-on capable terminal being installed.

# Chapter 79. Identity propagation

CICS Transaction Gateway can pass user security identity information (a distributed identity) from a JEE client in IBM WebSphere Application Server across the network to CICS Transaction Server for IBM 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 IBM WebSphere Application Server with security information in CICS Transaction Server for IBM z/OS.

CICS Transaction Gateway supports identity propagation for JEE client requests from IBM WebSphere Application Server to CICS Transaction Server for IBM 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 Chapter 105, "Request monitoring exits," on page 519 for more information.

## Benefits of using identity propagation

Identity propagation provides end-to-end security and consistent accountability, when applications in IBM WebSphere Application Server are connected to CICS.

Identity propagation provides the following benefits:

- An end-to-end solution for security when connecting IBM WebSphere Application Server to CICS Transaction Server for IBM 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 IBM WebSphere Portal. For more information, see the documentation for IBM WebSphere Application Server.
- "Single sign-on" authentication of users in IBM WebSphere Application Server before they are authorized in CICS Transaction Server for IBM 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 IBM WebSphere Application Server supported by CICS Transaction Gateway. For more information, see "JEE application servers" on page 30.
- Any user registry that is supported by IBM WebSphere Application Server. For more information, see the documentation for IBM WebSphere Application Server.
- CICS Transaction Server for IBM z/OS Version 4.1 (with APAR PK83741 and APAR PK95579), or later. For more information, see the CICS Transaction Server for IBM z/OS information center.

- IBM z/OS Version 1.11 or later.
- IBM RACF Security Server for IBM z/OS Version 5 or later. For more information, see *Introduction to CICS Security with RACF* in the CICS Transaction Server for IBM z/OS documentation.

### Network topology for using identity propagation

Identity propagation is supported only on IPIC connections to CICS configured with SSL.

For more information about the topologies that are supported by CICS Transaction Gateway, see Chapter 4, "Deployment topologies," on page 11.

The following example shows identity propagation in a topology with CICS Transaction Gateway in local mode:

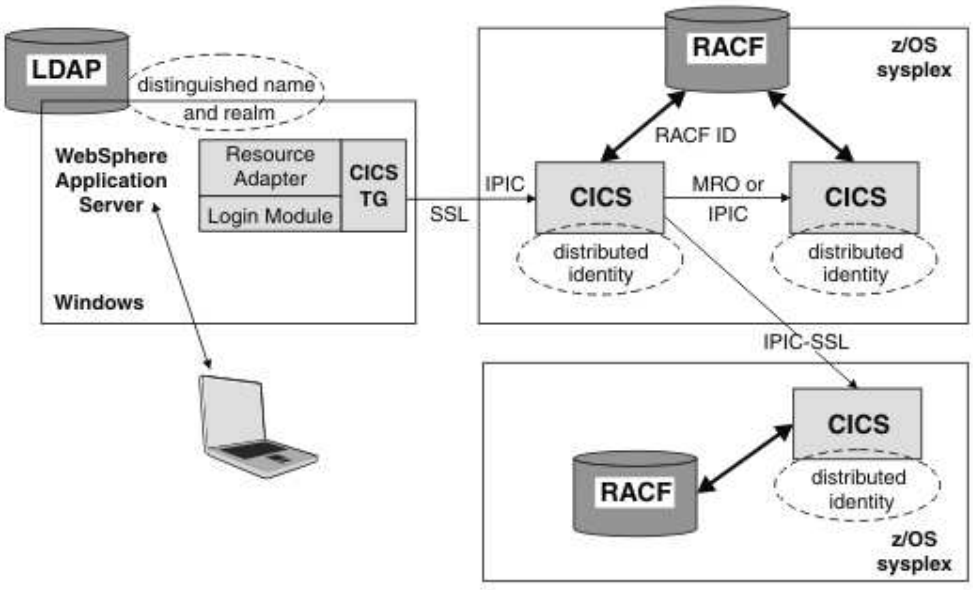



*Figure 29. Example of identity propagation in a local mode topology*

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 IBM 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.

# Part 11. Performance

The performance of individual components, including CICS Transaction Gateway, can affect overall system performance.

You can find more information about CICS TG performance in the CICS Transaction Gateway Performance Reports.

**Related reference**:

"List of statistics" on page 533
These statistics are available from the CICS Transaction Gateway.

**Related information**:

"Displaying statistics" on page 528
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.

# Chapter 80. 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 provide an understanding of which system components most affect performance and include the following information:

- Processor loading
- Data transfer rates
- Response times

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
- CICS Transaction Gateway
- CICS servers

The performance of CICS Transaction Gateway also depends on whether:

- 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:

- Request monitoring exits

- Performance monitoring tools such as RMF™ (Resource Management Facility), request monitoring exits, and IBM Tivoli OMEGAMON XE for CICS (on IBM z/OS)
- Statistics for monitoring and managing system resources

For more information, see Chapter 7, “Statistics and monitoring,” on page 19

# Chapter 81. Tuning the Gateway

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 157 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 157 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 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 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 158 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.

## Display TCP/IP host names

Selecting this option might cause severe performance reduction on some systems. See "Display TCP/IP host names" on page 168.

## 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 Chapter 38, "Configuring Gateway daemon settings," on page 157 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 167.

**Related reference**:

"List of statistics" on page 533
These statistics are available from the CICS Transaction Gateway.

**Related information**:

"Displaying statistics" on page 528
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.

## 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 threads 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 threads are allocated to run requests from remote Clients. When a worker thread finishes 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 Chapter 38, "Configuring Gateway daemon settings," on page 157 for information on setting configuration parameters.

The following table shows thread limits that must be considered when setting the number of connection manager and worker threads on the various operating systems:

*Table 30. 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 on Itanium | 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 (see your Linux Threads documentation for more information) |
| Solaris on SPARC | No limit | No limit |
| Windows | No limit | Limited by the amount of virtual memory available for the process (by default a thread has 1 MB of stack meaning that 2028 threads can be created per process) |

The threading model is illustrated in the following figure:

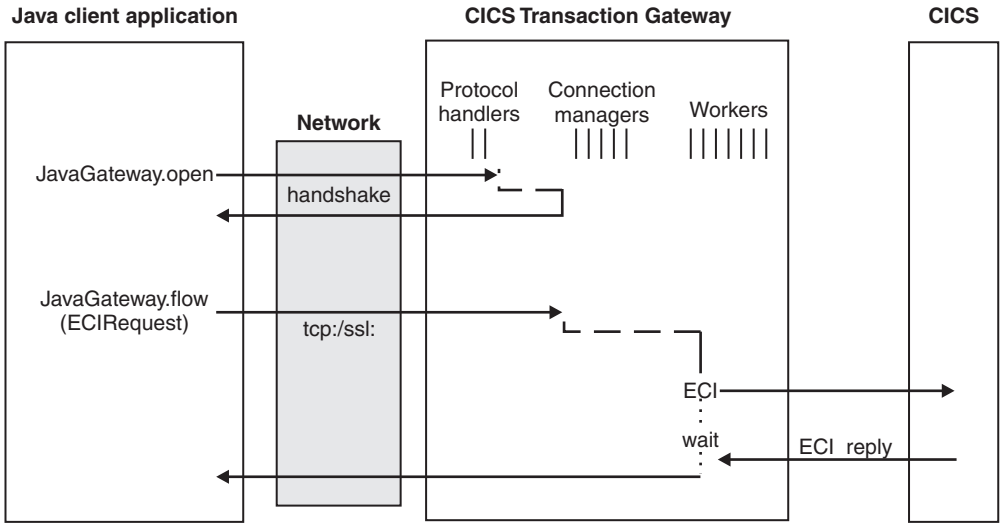



*Figure 30. CICS Transaction Gateway Threading model for TCP/IP and SSL protocols using a persistent socket*

**Related reference**:

"List of statistics" on page 533
These statistics are available from the CICS Transaction Gateway.

**Related information**:

"Displaying statistics" on page 528
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.

# Chapter 82. Tuning the Gateway to avoid out-of-memory conditions

The CICS Transaction Gateway statistics provide information to help you avoid out-of-memory conditions.

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

# Chapter 83. Tuning the JVM

Performance considerations related to Java include the size of the Java heap, whether JIT (Just In Time compiler) is enabled, and whether client applications are using persistent connections.

## Maximum heap size

If your system requires large numbers of connection manager threads you might need to increase the heap size to improve performance.

"System environment statistics" on page 544 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 the JIT is enabled; it is enabled by default. Immediately after a CICS starts, performance might be relatively slow because of JIT overheads. See your JVM documentation for information about JIT techniques.

## JavaGateway objects

Performance is better if you flow multiple requests using the same JavaGateway object than if you create a 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, and garbage collection.

## Client connections

Performance is improved if client applications that flow multiple requests to the CICS TG use the same connection for all of the requests. Whenever a connection is closed and reopened there is an overhead of cleaning up the resources from the old connections and allocating new resources for the new connection.

# Chapter 84. Configuring Windows for high network connection rates

When a TCP/IP socket is closed, it goes into TIME_WAIT state before closing, for a period of time determined by the operating system. A socket in TIME_WAIT state cannot be reused; this can limit the maximum rate at which network connections can be created and disconnected.

The Java Client application normally closes the socket; if the application is on a different machine from the CICS Transaction Gateway, the limitation usually applies to the machine running the application. The symptoms of a machine that is reaching these limits include:

- All of the TCP/IP resources of the operating system are in use, and requests for new connections fail. This causes `JavaGateway open` requests to fail intermittently, and throw a java.net.BindException.
- Running the `netstat -a` command on the application machine shows a large number of sockets in TIME_WAIT state.
- Performance deteriorates.

To improve the ability of the Windows operating system to deal with a high rate of network connections, add the following registry entries in

`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TCPIP\Parameters`

**`TcpTimedWaitDelay`**
: A DWORD value, in the range 30–300, that determines the time in seconds that elapses before TCP can release a closed connection and reuse its resources. Set this to a low value to reduce the amount of time that sockets stay in TIME_WAIT.

**`MaxUserPort`**
: A DWORD value that determines the highest port number that TCP can assign when an application requests an available user port. Set this to a high value to increase the total number of sockets that can be connected to the port.

For example, a system making a large number of connection requests might perform better if **TcpTimedWaitDelay** is set to 30 seconds, and **MaxUserPort** is set to 32678. See the operating system documentation for more details.

# Chapter 85. IPIC considerations

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.

When running with 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 Chapter 82, “Tuning the Gateway to avoid out-of-memory conditions,” on page 419 for more information.
- Increase the stack available to the Java threads using the **-Xss** parameter. The default value of -Xss is 512 KB with Java V7 64-bit. 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 Server Diagnostics reference.

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 (128 MB) is adequate for an SNA or TCP/IP workload, it might be necessary to increase this to 256 MB or 512 MB for an IPIC workload.

# Chapter 86. Client applications

The parameters you set for your application can affect performance.

## 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: Developing Applications* 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.

## 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.

# Chapter 87. Tracing

Full tracing of CICS Transaction Gateway can degrade system performance and should 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 501. 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.

# Chapter 88. 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 Windows, UNIX and Linux operating systems.

### vmstat

This tool provides statistics about virtual memory, disks, traps, and processor activity. Use this tool to determine system loading.

### iostat

This tool, available on UNIX and Linux, provides processor statistics and I/O statistics for tty, disks, and CD-ROM drives.

### sar (system activity report)

This tool, available on UNIX and Linux, collects, reports, or saves system activity information.

### IBM Performance Toolbox

This toolbox, available on UNIX and Linux, provides a set of useful performance tools that are integrated into a graphical user interface.

### perfmon (performance monitor)

This tool is available on Windows operating systems. For more information see the documentation for your operating system.

### netstat

This tool shows network status.

Refer also to the performance and tuning documentation for IBM WebSphere, SNA, TCP/IP, CICS Transaction Server for IBM z/OS, and IBM TXSeries and the documentation supplied with your operating system.

# Chapter 89. Statistics and performance assessment

Use statistics to assess system performance and to identify and resolve performance problems.

Performance problems are indicated by poor response times. 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.

You can use the statistics that are generated by CICS Transaction Gateway to establish the reasons why response times might be poor. The following statistics contain advice for improving poor response times.

### Investigate slow transaction response times in CICS

Slow transaction processing times in CICS cause increased response times. This can occur when a CICS system becomes constrained, or when interconnected database systems cause delays in transaction processing.

Investigate CICS response times using CICS monitoring facilities and resolve any constraints that you find. Consider the setting of MAXTASK and TRANCLASS CICS server parameters.

Monitor the CICS TG statistics CS_IAVRESP and CSx_IAVRESP for each CICS server. If the value of CS_IAVRESP is higher than you anticipate for the transaction, a CICS server might be constrained, or interconnected database systems might be causing delays in transaction processing.

### Worker thread queuing in the Gateway daemon

Transaction requests queue in the Gateway daemon due to high usage of worker threads. This can occur when the number of allocated connection managers is greater than the number of available worker threads.

You can establish whether your worker threads have been queuing this is the case by considering the value in WT_ITIMEOUTS. If `WT_ITIMEOUTS > 0` worker threads have been queuing. Also you can establish whether all your worker threads are in use by considering the value in WT_CCURR. If `WT_CCURR = WT_CALLOC` all worker threads are in use.

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. 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

An insufficient number of configured connection managers in the Gateway daemon causes I/O errors in a remote client. This can occur when all available connection manager threads are allocated to remote clients.

If `CM_CALLOC = CM_SMAX` all available connection manager threads are allocated to remote clients.

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

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. This can occur when large payloads, such as 32 KB COMMAREAs, are transmitted without the network payload being optimized using null truncation.

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. If so 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

The transmission of large amounts of data causes increased response times due to network latency over the connection between CICS Transaction Gateway and CICS. This can occur when large payloads, such as 32 KB COMMAREAs, are transmitted without the network payload being optimized using null truncation.

If `GD_IAVRESP - CS_IAVRESP = a high value` there might be constraints in the network between CICS Transaction Gateway and CICS. If so 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

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. 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.

You can establish whether Gateway processing time is high by using the statistics GD_IAVRESP and CS_IAVRESP. If `GD_IAVRESP-CS_IAVRESP > 100 milliseconds` Gateway processing time is high.

You can establish whether connection managers are queuing for worker threads by using the statistics CM_IALLOCHI and WT_IALLOCHI. If `(CM_IALLOCHI > WT_IALLOCHI) and WT_IALLOCHI > 0` connection managers are queuing for worker threads.

You can establish whether JVM garbage collection (GC) is constrained by using the following statistics SE_CHEAPGCMIN, SE_SHEAPMAX, SE_IGCTIME, GD_IRUNTIME and SE_IGCCOUNT. If:

```
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
```

If any of these three conditions are true the JVM GC is constrained. If so, increase Gateway daemon minimum and maximum JVM heap sizes.

**Note:** SE_IGCTIME is measured in milliseconds and GD_IRUNTIME is measured in seconds.

# Part 12. High availability

High availability of connections to CICS servers is achieved using dynamic server selection (DSS), enabling the choice of CICS server to be determined at runtime based on actual server availability. Dynamic server selection also provides 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.

For deployments that use local mode IPIC connections to CICS from IBM WebSphere Application Server, the ECI connection factory can benefit from the automatic fail over capability that is provided in IBM WebSphere Application Server. The automatic fail over and fail back capability is supported when connected to CICS TS V4.1 and later. For more information, see "Configuring automatic fail over in IBM WebSphere Application Server" on page 458.

# Chapter 90. Default CICS server

The default CICS server is used for requests that do not specify a CICS server name.

The default CICS server is a product-wide setting. For information about configuring the default server, see “Default server” on page 213.

# Chapter 91. 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**:

Creating a CICS request exit

Configuring a CICS request exit
The **cicsrequestexit** parameter specifies the class used to perform dynamic CICS server selection for ECI requests and ESI requests.

CICS request exit programming reference

# Chapter 92. Windows Workload Manager

On Windows, use the Workload Manager to dynamically select servers and balance the workload in your system effectively. The Workload Manager is used exclusively for connections to CICS servers over TCP/IP or SNA.

For information on configuring the Windows Workload Manager see "Configuring the Windows Workload Manager" on page 213.

## Workload Manager overview

The Workload Manager handles various problems when the Client daemon is used to connect with one or more CICS servers.

- How to distribute work across available, equivalent CICS servers.
- How to favor one CICS server over another (*biasing*).
- How to handle requests that fail to reach a target CICS server.

The Workload Manager resolves these problems and others by dynamically selecting target CICS servers based upon criteria that you configure. It ensures that all calls in an extended Logical Unit of Work (LUW) are routed to the same CICS server.

The Workload Manager cannot be used with servers that are connected using the IPIC protocol.

The Workload Manager uses the Windows Registry during normal operation. Any user ID using the Workload Manager must have sufficient user permissions to access and update the following registry key and any subkeys: `HKEY_LOCAL_MACHINE\SOFTWARE\IBM\CCLX`. On a Windows 64-bit system this will appear in `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\IBM\CCLX` The installer creates the registry key with security defined for standard users to update the registry key.

### Key concepts

An alphabetic list of key concepts that it is essential to understand.

**Programs**
: CICS programs are executable units that can be started within a CICS server. A particular CICS program might be available on multiple CICS servers. The Workload Manager has a list of CICS servers on which the program is defined (*program affinity*).

**Server group**
: The Configuration Tool uses server groups to group together server instances. A Server Group can correspond to one or more CICS servers.

  The first time the Client daemon makes a request, it selects a server instance from the group at random. All subsequent requests to that Server Group are sent to the same server instance, until that server becomes unavailable, when another server from the group is selected.

An ECI program can be associated with a Server Group. The Workload Manager assumes that the program exists on all server instances in the Server Group.

**Workload management**
The ability of the Workload Manager to select a suitable CICS server as a target for a client request. The load balancing algorithms try to spread the workload over various server instances on a particular CICS server (via random selection) and they allow the user to effectively specify a bias for the selection of a particular CICS server. Known inactive CICS servers are not considered for selection.

## Information required by Workload Manager

The Workload Manager must know the names of the server groups between which workload can be managed, and the names of the CICS servers that belong to each server group. The Workload Manager can also (optionally) know the names of the programs that can run in each server group.

You supply this information using the Configuration Tool which produces the appropriate entries in the configuration file. For more information see "Configuring the Windows Workload Manager" on page 213.

The following table shows the information that the Workload Manager requires. To enable workload management the Client daemon must be able to correlate CICS servers against the server groups to which they belong.

| Server group | CICS server |
|---|---|
| GROUP1 | CICS1<br>CICS2<br>CICS3 |
| GROUP2 | CICS4<br>CICS5 |
| GROUP3 | CICS6 |

The Client daemon can connect to a specific server group through only one CICS server. So, in the example, a Client daemon can connect to a server group using one of the following groups of servers:

- CICS1, CICS2, CICS3
- CICS4, CICS5
- CICS6

For example, connections from a specific client to CICS2, CICS4, and CICS6 would be valid but connections to CICS1, CICS3, and CICS5 would not be valid because CICS1 and CICS3 belong to the same server group.

Not all CICS programs need be available on all possible server groups. The Workload Manager knows the name of the program that the client request is for, and can select, from a subset of all server groups, those server groups on which a

particular program is defined. The Workload Manager also has knowledge of a set of server groups where undefined programs can be routed.

For example:

| Program | Server group |
|---|---|
| PROG1 | GROUP1<br>GROUP2<br>GROUP3 |
| PROG2 | GROUP1<br>GROUP3 |
| PROG3 | GROUP2 |

When the Workload Manager receives requests for programs which have not been defined in the configuration file, the round robin algorithm distributes the requests across all server groups and the biasing algorithm uses the default server.

## Load balancing algorithms

You can select a *round-robin* algorithm or a *biasing* algorithm for load balancing.

You select an algorithm using the Configuration Tool; see "Configuring the Windows Workload Manager" on page 213.

### The round-robin algorithm

The round-robin algorithm assumes that all server groups are equally valid for selection.

In the round-robin algorithm, when the Client daemon is initially started, it reads from the configuration file a list of all possible server groups to which any ECI or EPI request can be sent.

The Workload Manager also records the last server group selected. When a new ECI or EPI request is made, the next server group in the list is selected as the target. When it reaches the last server group it loops around to the first one.

#### Examples

For each of these examples, assume that workload management is set up with the example configuration described in "Information required by Workload Manager" on page 444, and that workload management is enabled for all programs and server groups.

- The first request sent for the program PROG2 goes to GROUP1, the second request for PROG2 goes to GROUP3, the third to GROUP1 again, and so on.
- All requests to PROG3 go to GROUP2.
- Half of the requests to PROG2 go to GROUP3, and all of these requests go to CICS6 because this is the only server in GROUP3.

Thus, the first request for PROG2 is sent to GROUP1, and a server from GROUP1 is selected at random. For example, CICS2 might be selected. The second request for PROG2 is sent to GROUP3, and CICS6 is selected because it is the only server in GROUP3. The third request for PROG2 is sent to GROUP1 again, and CICS2 is selected because it was the first server selected in GROUP1.

### The biasing algorithm

The biasing algorithm provides a way of balancing workload by specifying that workload distribution should favor particular server groups. For example, if there are two server groups with a bias of 75 and 25, program requests are sent in a ratio of 3:1 to the first server group.

If a server group fails, the internal biasing calculation changes. If two server groups are available, one with a bias of 100 and the other with a bias of 0, all requests are sent to the first server group. If the first server group becomes unavailable, all requests are directed to the second server group. A bias value of 0 is a special case, meaning *use only if no other server group is available*.

The biasing algorithm works only for ECI calls. If you try to run an EPI application whilst the biasing algorithm is selected, the round-robin algorithm is used instead.

#### Examples

For each of these examples, assume that workload management is set up with the example configuration described in "Information required by Workload Manager" on page 444, and that workload management is enabled for all programs and server groups.

- For PROG1, if GROUP1 has a bias of 6, GROUP2 has a bias of 2 and GROUP3 has a bias of 2, approximately 60% of requests for PROG1 go to GROUP1, and approximately 20% to each of GROUP2 and GROUP3. The selection of servers within each group is random, but the Client daemon will only ever use one server from each group.
- For PROG2, if GROUP2 has a bias of zero, and GROUP3 has a bias of 1, all requests are sent to GROUP3 (and therefore CICS6). If CICS6 becomes unavailable, all requests are sent to GROUP2.

## Workload Manager failure recovery

If a request to route an ECI call to a particular CICS server fails, indicating that the server is no longer available, the server is removed from the list of available servers.

If a request fails to reach the target CICS server the Workload Manager tries again, this time sending it to a different server that it knows is available. The server that failed is removed from the list of active servers. It is added back after a period of time which you specify in the `Server group timeout` field in the Configuration Tool. If the server fails again, the process is repeated.

## Workload Manager implementation

The Workload Manager is invoked during ECI and EPI calls from the Client daemon, and implements ECI and EPI user exit functions.

These ECI and EPI exit functions are accessed during processing at a number of points during the execution of a request. They are described in *CICS Transaction Gateway for Multiplatforms: Developing Applications*.

The exit is implemented as a re-entrant multi-threaded Dynamic Load Library (DLL). This exit (code) is dynamically loaded by each client application that executes. The DLL executes in the process address space of the ECI-calling application rather than the Client daemon.

Instead of using the supplied Workload Manager you can write your own user exits that implement the ECI and EPI functions, and these exits can be driven by the cicsterm and cicsprnt commands. Information on producing such exits is given in the *CICS Transaction Gateway for Multiplatforms: Developing Applications*.

When the Workload Manager is activated for a cicsterm or cicsprnt command, it determines which server the Client daemon should connect to. The Workload Manager server selection overrides any server specified with the cicsterm or cicsprnt command.

## ECI implementation

The Workload Manager is implemented as a Windows DLL called `cclmecix.dll`. This DLL contains implementations for the following listed functions.

**CICS_EciInitializeExit**
Initialize the exit. Called once per process at the first ECI call. This function builds the list of available server groups and other attributes.

**CICS_EciExternalCallExit1**
Called at the start of an ECI call. This function is passed a reference to the ECI parameter block and can change the value of the **`eci_system_name`** field to select an alternate CICS server. The value of this field can be a Server name. This function selects the correct target CICS server for each ECI client request. This function involves workload management algorithms.

**CICS_EciSystemIdExit**
If on return from the `CICS_EciExternalCallExit1`, the ECI request fails to reach the target CICS server, either because of communication failure or a CICS server failure, `CICS_EciSystemIdExit` is called to select an alternate CICS server target. The function flags that a CICS server is no longer available and should be removed from the list of selectable CICS servers. The workload management algorithm is then re-executed to select the next CICS server to be targeted by the Client daemon.

## EPI implementation

The Workload Manager is implemented as a Windows DLL called `cclmepix.dll`. This DLL contains implementations for the following listed functions.

**CICS_EpiInitializeExit**
Initialize the exit. Called once per process at the first EPI call. This function builds the list of available regions and other attributes.

**CICS_AddTerminalExit**
Selects a CICS server.

**CICS_EpiSystemIdExit**
`CICS_EpiSystemIdExit` is called to select an alternate CICS server target. The function flags that a CICS server is no longer available and must be

removed from the list of selectable CICS servers. The workload management algorithm is then re-executed to select the next CICS server to be targeted by the Client daemon.

## Workload Manager installation

The ECI Exit DLL (`cclmecix.dll`) and the EPI Exit DLL (`cclmepix.dll`) are installed as part of CICS Transaction Gateway.

The exits are used by ECI and EPI client calls without modification to any existing ECI or EPI program.

The supplied load management DLLs can coexist with any user-written DLLs. With the exception of **CICS_EciSetProgramAliasExit**, user-written DLLs are called after the load management DLLs.

## Tracing the Workload Manager

Tracing of the Workload Manager is not enabled by default. You can enable Workload Manager tracing in the following two ways.

1. Select Workload Manager in the **Trace** section of the Configuration Tool.
2. Specify **LMG** on the cicscli -m command.

   This command overrides the configuration setting.

# Part 13. Deploying applications

The method you must use to deploy your application depends on the CICS TG API used by the application.

# Chapter 93. Deploying a CICS resource adapter

The resource adapters are provided as standard modules, ready for deployment into a Java Platform, Enterprise Edition (JEE) application server. The resource adapters can be packaged in a JEE application along with other components such as Enterprise JavaBeans, 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 31-bit and 64-bit run time environments. For more information on supported environments, see "JEE application servers" on page 30.

If installing both the `cicseci.rar` and `cicsepi.rar` into the same IBM WebSphere Application Server, then the resource adapters should be isolated from each other by enabling the configuration option when each resource adapter has been installed.

For information on how to deploy a CICS resource adapter 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 Java 2 Security permissions are enabled on your JEE application server, consider setting the security permissions that allow CICS Transaction Gateway to access your keystores. For more information, see the *CICS Transaction Gateway: Developing Applications*.

## 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 adapter 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:`.

**interceptPlugin**
The class name of an intercept plug-in that can be used to test JEE components without a running CICS Transaction Gateway or CICS Transaction Server. For more information see the CICS Transaction Gateway Programming Guide.

**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 239. 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 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. If a ClientSecurity class is specified, an equivalent ServerSecurity class must be specified on the serverSecurity parameter. 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: Developing Applications*.

**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. If a ServerSecurity class is specified, an equivalent ClientSecurity class must be specified on the **clientSecurity** parameter. 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: Developing Applications*.

**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 507.

**cipherSuites**
The cipherSuites parameter 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: Developing Applications*.

**ipicSendSessions**
In local mode, this parameter sets the maximum number of simultaneous transactions, or CICS tasks, that are allowed over the connection. The actual number of send sessions used is determined by the connection factory property, or the IPCONN RECEIVECOUNT parameter in CICS Transaction Server for IBM z/OS, whichever is lower.

**ipicHeartbeatInterval**
In local mode, this parameter sets the time in seconds that an IPIC connection must be inactive before heartbeats are sent to the CICS server. Enter a value in the range 1 to 3600, or 0 to disable IPIC heartbeats.

**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.

## Predefined attributes

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

**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 adapter has 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.

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

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

**Server retry interval**
Server retry interval is disabled for local mode IPIC connections. Connection attempts are only initiated if a request is directed at the CICS server and no connection attempt is already in progress.

## EPI resource adapter deployment parameters

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:`.

**interceptPlugin**
The class name of an intercept plug-in that can be used to test JEE components without a running CICS Transaction Gateway ( CICS TG) or CICS server. For more information see ../progguide/topics/intercepting_eci_enable_jee.dita

**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 213). 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 JEE Connector Architecture in the *CICS Transaction Gateway for Multiplatforms:Developing Applications* 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: Developing Applications*.

**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: Developing Applications*.

**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 404.

**encoding**
The Java Encoding to use when creating 3270 data streams. The encoding is converted to the appropriate CCSID. See Chapter 109, "Supported conversions," on page 557 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: Developing Applications* 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 JEE 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 Chapter 102, "Tracing," on page 499.

**cipherSuites**
The cipherSuites parameter can be used when establishing an SSL connection. In the IBM 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.

## Predefined attributes

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 JEE/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).

# Configuring automatic fail over in IBM WebSphere Application Server

ECI connection factories that are configured to use a local mode IPIC connection to CICS can benefit from the automatic fail over capability that is provided in IBM WebSphere Application Server.

## Before you begin

The automatic fail over and fail back capability is supported when connected to CICS TS V4.1 and later.

## About this task

IBM WebSphere Application Server allows a connection factory to be configured with the JNDI name of an alternative connection factory that is configured to use a different CICS server. When a failure occurs on a connection to the primary CICS server, new transactions are automatically routed to use the alternative connection factory. When the connection to the primary CICS server becomes available, new transactions are automatically routed to use the primary connection factory.

## Procedure

1. Create and configure the primary ECI connection factory with a local mode IPIC connection to CICS.
2. Create and configure an alternative ECI connection factory with the same configuration as the primary connection factory, but with a different IPIC connection URL in the `serverName` property.
3. Add the `alternateResourceJNDIName` custom property to the connection pool properties for the primary connection factory. Set the property value to the JNDI name of the alternative connection factory.

   For more information, see Resource workload routing in the IBM WebSphere Application Server documentation.

# Chapter 94. Deploying remote Java applications

Remote Java client applications are deployed to the runtime environment as Java Archive (`.jar`) files.

You are licensed to copy the following files to the computer that is running the Java client application:

- For non-JEE applications, copy the file `ctgclient.jar`.
- For JEE applications in a managed environment, copy the resource adapters (RAR files) in the `<install_path>/deployable` directory.
- For JEE applications in a nonmanaged environment, copy the following files in the `<install_path>/deployable/classes` directory:

```
cicsjee.jar
ctgclient.jar
ccf2.jar
screenable.jar
```

Ensure that any JAR files that you copy are listed on the class path of the remote computer.

### 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` directory. The JEE resource adapters are in the `<install_path>/deployable` directory.

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.

## Java options for the Solaris JVM

You should 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 become unresponsive when either the TCP/IP or the SSL protocol is used. See your Solaris on SPARC documentation for more details.

# Chapter 95. 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 a 32-bit version of this file in the `<platform>/lib` directory of the SDK package. This file can also be found in the `<install_path>/lib` directory of an installed CICS TG.

You can find a 64-bit version of this file in the `<platform>/lib64` directory of the SDK package. This file can also be found in the `<install_path>/lib64` directory of an installed CICS TG.

On IBM AIX: `libctgclient` in `<platform>/lib` supports both 32-bit and 64-bit operation. There is no `<platform>/lib64` directory.

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.

## ECI V2 and ESI V2 error log

To enable the error logging of system errors, socket errors, and other Gateway connection errors, turn on CTG_TRACE_LEVEL1 trace. For more information see "Tracing in ECI V2 and ESI V2 applications" on page 505.

## CICS Transaction Gateway Desktop Edition:

The license does not permit redistributing components. Each system which you install CICS Transaction Gateway Desktop Edition on or any of its components must have a separate license.

# Chapter 96. Deploying Microsoft NET Framework-based applications to remote systems

Remote NET Framework-based applications are deployed to the Windows runtime environment as an executable assembly (`.exe`) or library assembly (`.dll`), depending on the type of application.

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

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

The CICS Transaction Gateway API for .NET Framework supports 32-bit and 64-bit operation. Support is provided by a single assembly (`IBM.CTG.Client.dll`) which is included in the SDK 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 Framework-based application.

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

**CICS Transaction Gateway Desktop Edition:** The license does not permit redistributing components. Each system on which you install CICS Transaction Gateway Desktop Edition on or any of its components must have a separate license.

# Part 14. 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 as described on the support page at CICS Transaction Gateway Support Portal.

# Chapter 97. 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 253. You can also use the "JCA resource adapter installation verification test (IVT)" on page 251 to test that the connection from IBM 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 Part 5, "Upgrading," on page 75.

## What messages were produced about the problem?

CICS Transaction Gateway writes information, warning and error messages to the message logs (for more information, see Chapter 101, "General information about messages," on page 497). 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 Part 6, "Configuring," on page 91.

## 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 Chapter 106, “Statistics,” on page 523.

Regular backup jobs or other system maintenance might also cause unexpected problems at specific times of day.

# Chapter 98. 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 documentation. 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 Chapter 100, “Dealing with problems,” on page 475 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 Chapter 99, “Problem determination tools,” on page 471 or you might need to refer to Chapter 103, “Problem solving and support,” on page 511.

# Chapter 99. 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 additional information on Java diagnosis see: . `http://publib.boulder.ibm.com/infocenter/java7sdk/v7r0/index.jsp`

For more information about trace see Chapter 102, "Tracing," on page 499.

## Java diagnostic tools

Links are provided in the online information on Java diagnostic tools.

Links are provided in the online information on Java diagnostic tools. For additional Information on Java diagnosis, see: . `http://publib.boulder.ibm.com/infocenter/java7sdk/v7r0/index.jsp`

### Java diagnostic output

When starting CICS Transaction Gateway on Windows, any standard output or error text produced is captured in the file `jvmdbg.log`, located in the `<product_data_path>` directory. The purpose of this file is to capture debug output from the jvm, when the CICS Transaction Gateway is started with debug java startup parameters specified. For example:

```
ctgservice -R -A-j-verbose:init
```

### 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 346.

With some Java Virtual Machines (JVMs) on UNIX and Linux you can force Java to write a stack dump showing the states of the current threads.

For example, on IBM JVMs, 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.

## 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.

Refer to your SNA communications product documentation for information on the APING utility.

## 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 346.

With some Java Virtual Machines (JVMs) on UNIX and Linux you can force Java to write a stack dump showing the states of the current threads.

For example, on IBM JVMs, 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.

## 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. (Linux operating systems only.) |
| ipconfig | Display all TCP/IP network configuration values. This is useful when determining whether or not an IP interface is active. (All operating systems except Linux.) |
| 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. |

| Command | Purpose |
| --- | --- |
| ping | Verify connection to a remote computer or computers. The equivalent command for IPv6 is ping6. |
| tracert | Trace TCP/IP path to a requested destination. This is useful for determining whether a problem exists with an intermediate node or not. The equivalent command for IPv6 is tracert6. (Windows operating systems only.) |
| traceroute | Trace TCP/IP path to a requested destination. This is useful for determining whether a problem exists with an intermediate node or not. (All operating systems except Windows.) |

# Chapter 100. 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 IBM 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 Windows" on page 60 and "Location of the installation logs on Unix and Linux" on page 62.

#### 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 when using IBM AIX 7.1

Installation of CICS Transaction Gateway on IBM AIX 7.1 can fail if Service Pack (SP) 2 is not applied.

#### Symptom

The installer exits without installing CICS Transaction Gateway, and the following message is written to the console:

```
Error: Port Library failed to initialize
Could not create the Java virtual machine.
```

#### Probable cause

Your IBM AIX operating system is IBM AIX 7.1 Technology Level (TL) 1, Service Pack 0 or 1.

### Action

To determine the level of your operating system, you can run the following command.

```
oslevel -s
```

You will see a result like

```
7100-01-01-1141
```

This is TL 1, SP 1. The second number is TL. The third number is the SP.

Upgrade IBM AIX 7.1 to SP2 or later.

## 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.
```

### 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.

Details of the running processes are written to `cicstgFind.log`.

For more information see “Location of the installation logs on Windows” on page 60 and “Location of the installation logs on Unix and Linux” on page 62.

## Insufficient disk space

An unattended installation might fail due to insufficient disk space being available.

### Symptom

On Windows the installer fails and the following message is displayed on the console (applies to all install modes):

```
An error occurred during installation.
```

There are no other messages displayed on the console or the install log which indicate the cause of the failure.

On UNIX and Linux the installer fails to launch and the following message is displayed:

```
WARNING: /tmp does not have enough disk space!
         Attempting to use / for install base and tmp dir.

WARNING! The amount of / disk space required to perform
this installation is greater than what is available.  Please
free up at least XXXXX kilobytes in / and attempt this
installation again.  You may also set the IATEMPDIR environment
variable to a directory on a disk partition with enough free
disk space.  To set the variable enter one of the following
commands at the UNIX command line prompt before running this
installer again:

- for Bourne shell (sh), ksh, bash and zsh:

     $ IATEMPDIR=/your/free/space/directory
     $ export IATEMPDIR

- for C shell (csh) and tcsh:

     $ setenv IATEMPDIR /your/free/space/directory
```

### Probable cause

If there are no other messages specifying the cause of the failure, the most probable cause is that there is insufficient disk space is available to launch the installer or install the product.

### Action

Free up some space and relaunch the installer.

For more information on installation disk space requirements see Chapter 19, “Preparing to install CICS Transaction Gateway,” on page 55.

## Startup and shutdown problems

Problems when starting and stopping CICS Transaction Gateway.

### Gateway daemon and Client daemon fail to start on 64-bit Linux

A Gateway daemon fails to start on a 64-bit Linux operating system.

#### Symptom

Startup fails with errors such as the following:

- **cicscli** error while loading shared libraries: libncurses.so.5: cannot open shared object file: No such file or directory.
- 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)'`

#### 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.

## 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

Outstanding logical units of work from ECI or EPI requests that are waiting to complete, prevent the Gateway daemon from quiescing.

### Action

Wait for the API calls to complete. 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)

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 Library at: `http://www-01.ibm.com/software/htp/cics/library/`.

If normal shutdown fails you can promote this to an immediate shutdown.

## Statistics errors in the Client daemon log for UNIX and Linux

The Client daemon has shut down before the Gateway daemon.

### Symptom

When the Gateway daemon attempts to gather statistics, including when it attempts to shut down, the following messages might be written to the Client daemon error log:

```
05/24/12 12:00:00.931 [2277] STA:
   CCL1046E Error in function 'OsMPLockRequest' (Error Code = 2)
05/24/12 12:00:00.948 [2278] STA:
   CCL1046E Error in function 'OsMPLockFree' (Error Code = 2)
```

### Probable cause

The Client daemon has shut down before the Gateway daemon.

### Action

Perform an immediate shut down on the Gateway daemon. Always ensure that the Gateway daemon is shut down before the Client daemon.

## Message CTG8276E logged during startup

Message CTG8276E indicates that there is a problem with the security privileges.

### Symptom

Message CTG8276E `Insufficient security permissions to communicate with the CICS Transaction Gateway service` is logged when starting CICS Transaction Gateway.

### Probable cause

The **ctgadmin** command does not have sufficient privileges to interact with the CICS Transaction Gateway service running on Microsoft Windows.

On Windows platforms, CICS Transaction Gateway runs as a Microsoft Windows Service, and the **ctgadmin** utility uses the Windows Service Control Manager (SCM) to start and stop the product. Starting or stopping a Windows Service requires administrator permissions; this means that the **ctgadmin** command must be run with administrator permissions when starting or stopping CICS Transaction Gateway. As part of the Windows User Account Control (UAC) feature, applications are started with standard user account permissions for security reasons, even if the user has administrator privileges. To resolve this problem, when starting or stopping CICS TG, you must be logged in using a user with administrative permissions and you must start **ctgadmin** with the Windows `Run as Administrator` function. When using **ctgadmin** for tasks other than starting or stopping CICS Transaction Gateway, administrator permissions are not required.

### Action

See your Microsoft Windows documentation for more information about UAC:

- What is User Account Control?
- User Account Control: Use Admin Approval Mode for the built-in Administrator account

## Message CCL1046E logged on startup after upgrade

Message CCL1046E with error 22 indicates a problem with shared memory left by a previous installation of CICS TG.

### Symptom

After upgrading to a new version of CICS Transaction Gateway on UNIX or Linux, message CCL1046E is logged. For example:

```
STA:CCL1046E Error in function 'shmget'
   (Error Code = 22 PID = 0x41B4 TID = 0x7E35B6D0)
```

### Probable cause

Message CCL1046E with error 22 indicates that a previous installation of CICS TG did not shutdown properly and did not clean up a piece of shared memory.

### Action

Use the **ctgclean** utility to clean up the orphaned shared memory. You can download this from:: http://www-01.ibm.com/support/docview.wss?rs=1083&context=SSGMGV&context=SSZPSF&context=SSNQZF&dc=D400&uid=swg24019221&loc=en_US&cs=utf-8&lang=en

If this does not resolve the problem, then reboot the system.

## Configuration problems

Problems with the way that CICS Transaction Gateway is configured.

### Configuration Tool character display incorrect

Configuration Tool characters are incorrectly displayed.

#### 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

Refer to the documentation supplied with your operating system and JRE for information on which font packages are required for which locales.

### Using X-Window System clients

There are some known problems when using certain X-Window System clients to display cicsterm or the configuration tool screens.

#### Symptoms

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.

#### Probable Cause

The configuration of the X-Window System client.

#### Action

Refer to the X-Window System client documentation.

### Automatic transaction initiation (ATI) failure

Automatic transaction initiation against a Client daemon terminal does not work.

### Symptom

A request to initiate a transaction against a Client daemon terminal fails.

### Probable cause

Different communications products provide different SNA implementations for handling inbound attaches. For the Client daemon using IBM Communications Server, Microsoft Host Integration Server, and IBM Personal Communications, you must predefine the CRSR attach program to relay the information to the Client daemon. This is necessary so that the CICS server can perform automatic transaction initiation (ATI) against the Client daemon terminals.

### Action

Define CRSR to use program CCLCLNT.EXE. For more information, see "Configuring SNA on Windows" on page 142.

## 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 245.
- If you are using the Remote API Client (for SNA), check the local "SNA error log" on page 498.
- 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 SNA" on page 131 and "Defining SNA connections on CICS Transaction Server for z/OS" on page 134.

## 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.

## 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

The CICS 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 the CICS server is listening on. To check the port an IPIC TCPIPService is defined to listen on in CICS Transaction Server:

   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 169.

### 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

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 that your configuration is correct. For more information see Part 7, "Scenarios," on page 255.

## 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 109.

Alternatively you can enable autoinstall on the TCPIPService. For more information see `http://www-01.ibm.com/software/htp/cics/library/`.

## cicsterm or cicsprnt 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** or **cicsprnt -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** or **cicsprnt -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 or cicsprnt 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 375 and "**cicsprnt** options" on page 383.

## Microsoft Host Integration Server problem

Microsoft Host Integration Server does not have the appropriate security access.

### Symptom

The following message is written to the cicscli.log:

```
CCL4661E: SNA stack component was not loaded or was terminated,
APPC return code X'F004'
```

### Probable cause

You are using Microsoft Host Integration Server and the service does not have the appropriate security access.

### Action

Configure CICS Transaction Gateway with a Windows domain logon with the appropriate access to the HIS server.

# Security problems

Problems with security.

## SSL problems

SSL connection problems are reported to Java client applications via message CTG6651E.

```
CTG6651E Unable to connect to the Gateway daemon: [address = IP address ,
port = port ] [error ]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 235.

### Application receives an “access denied” exception

An application configured to connect to the Gateway daemon using SSL 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 = cicstgd2, 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. For more information, see Permissions to access the file system.

### Key ring name not recognized

During configuration, the incorrect use of the backslash (\) character as a parameter delimiter prevents the key ring file being recognized.

### Symptom

The SSL protocol handler fails to start because the keyring file cannot be found.

### Probable cause

Java interprets the backslash (\) character as a parameter delimiter; a backslash (\) character has been incorrectly used as a directory separator in the path name of the key ring file during configuration, and Java does not recognize the name.

### Action

Check the SSL protocol configuration values in the CICS Transaction Gateway configuration file (ctg.ini). Ensure that either forward slash (/) character, or double backslash (\\) characters are used as separators in the path names on all operating systems. For example:

On UNIX and Linux:

```
/mykeys/jsse/keystore.jks
\\mykeys\\jsse\\keystore.jks
```

On Windows:

```
c:/mykeys/jsse/keystore.jks
c:\\mykeys\\jsse\\keystore.jks
```

## 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 problems

Problems when using identity propagation.

## Identity propagation not supported

A security exception and message CTG9631E occur when a back-level CICS server that does not support identity propagation is 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 see the CICS Transaction Server documentations Library at: `http://www-01.ibm.com/software/htp/cics/library/`.

## 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 IBM 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 documentation 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 IBM 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 caused by insufficient memory being available.

## Memory use increases over time

The amount of memory used by the Gateway daemon might increase 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.

- 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.

### 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.

The amount of memory that Java allocates to each thread is set by using the -Xmso and -Xss options. Do not change the Java stack and native stack sizes from their default values.

For more information on thread limits see "Threading model" on page 417. For more information on Java memory allocation and JVM stack sizes, see the *IBM Java Diagnostics Guides Information Center*.

Also see Chapter 82, "Tuning the Gateway to avoid out-of-memory conditions," on page 419.

**Related reference**:

"List of statistics" on page 533
These statistics are available from the CICS Transaction Gateway.

**Related information**:

"Displaying statistics" on page 528
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 157
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.

# 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, and 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

- In remote mode topologies, increase the **SENDSESSIONS** value in the CICS Transaction Gateway configuration file (`ctg.ini`).

- In local mode topologies using JEE, increase the value of the **ipicSendSessions** property in the connection factory configuration.
- In local mode topologies using Java base classes, use the **CTG_IPIC_SENDSESSIONS** Java property to set the maximum number of IPIC send sessions.
- Increase the **IPCONN ReceiveCount** value in CICS.
- Increase the Java heap size.

For more information see "Configuring IPIC on CICS Transaction Server for z/OS" on page 112

# 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 346.

## Connection failure exception occurs when running under load

Some versions of Windows limit the number of open requests on a given socket.

### Symptom

An intermittent ConnectFailed exception occurs when running under load.

### Probable cause

Workstation versions of Windows restrict the number of open requests that can be queued against a socket.

### Action

The ideal solution is to use a higher performance Windows operating system. For information about supported platforms see "Operating systems" on page 27.

## "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: Developing Applications.*

## Unable to load class that supports TCP/IP

If Java attempts to use class files from the local file system, this contravenes security rules and generates an exception.

### Symptom

The following error occurs when running applications:

```
java.io.IOException: CTG6664E Protocol tcp not supported
```

### Probable cause

You are using a Web browser and CICS Transaction Gateway on the same workstation, and the ctgclient.jar and ctgserver.jar are referenced in the CLASSPATH setting.

Java searches the CLASSPATH environment variable before downloading classes across the network. If the required class is local, Java attempts to use it. However, use of class files from the local file system contravenes the application security rules, and generates an exception.

### Action

Edit the CLASSPATH setting to remove ctgclient.jar and ctgserver.jar.

# Application development problems

Problems with developing applications for CICS Transaction Gateway.

## Linux compiler problem

Linux compiler problem.

### Symptom

When compiling the C and C++ samples on a 64-bit Linux operating system, the samples might fail to build 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.

## 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 Framework-based 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 Part 16, "Data conversion," on page 553.

## System errors in local mode Java applications

Local mode Java applications might experience system errors when run in a 64-bit JVM.

### Symptom

When you run a local mode Java application, requests fail with one of the following errors:

```
- ECI_ERR_SYSTEM_ERROR
- ESI_ERR_SYSTEM_ERROR
- EPI_ERR_FAILED
```

### Probable cause

When you use a 64-bit JVM, local mode Java applications cannot send requests to CICS using the TCP/IP, SNA protocols, or perform list systems requests.

### Action

Make one of the following changes:

- Use a 32-bit JVM to run the local mode application.
- Ensure that the local mode application does not perform list systems requests, and that it uses the IPIC protocol for communication with CICS.
- Convert the application to run in a remote mode configuration that connects to CICS Transaction Gateway.

### More information

For more information about APIs and the environments where they can be used, see Chapter 14, "Which API can be used?," on page 41

## Windows build script gives error unresolved external symbol

Problem using a Visual Studio command prompt to compile the ECI and ESI V2 samples with the `.cmd` build script.

### Symptom

The link command fails with the following messages:

```
> link /LIBPATH:"C:\Program Files (x86)\IBM\CICS Transaction Gateway\lib64"
/DEBUG "C:\ProgramData\IBM\CICS Transaction Gateway\samples\c\esi_v2\ctgesib1.obj"
ctgclient.lib
/OUT:"C:\ProgramData\IBM\CICS Transaction Gateway\samples\c\esi_v2\ctgesib1.exe"
```

```
Microsoft (R) Incremental Linker Version 12.00.30501.0
Copyright (C) Microsoft Corporation. All rights reserved.

ctgesib1.obj : error LNK2019: unresolved external symbol
   _CTG_openRemoteGatewayConnection referenced in function _openGatewayConnection
ctgesib1.obj : error LNK2019: unresolved external symbol
   _CTG_closeGatewayConnection referenced in function _closeGatewayConnection
ctgesib1.obj : error LNK2019: unresolved external symbol
   _CTG_getRcString referenced in function _displayRc
ctgesib1.obj : error LNK2019: unresolved external symbol
   _CTG_listSystems referenced in function _selectServer
ctgesib1.obj : error LNK2019: unresolved external symbol
   _CTG_ESI_verifyPassword referenced in function _verifyUser
ctgesib1.obj : error LNK2019: unresolved external symbol
   _CTG_ESI_convertTime referenced in function _verifyUser

C:\ProgramData\IBM\CICS Transaction Gateway\samples\c\esi_v2\ctgesib1.exe :
   fatal error LNK1120: 6 unresolved externals
```

### Probable cause

The wrong Visual Studio environment is being used. This can happen when the build script requests a 64-bit compilation but was run from a 32-bit Visual Studio command prompt or a 32-bit compilation was requested from a 64-bit Visual Studio command prompt.

### Action

Open a new Visual Studio command prompt with the correct build environment and run the build script.

## JSON web services problems

Problems with JSON web services.

### Web service connection closes as soon as HTTP request is sent

When an application attempts to send a web service request to the Gateway daemon, the connection to the Gateway daemon is dropped if an incorrect port is used.

#### Symptom

The HTTP or HTTPS connection between the web service client application and the Gateway daemon is closed as soon as a web service request is sent. If connection logging is enabled, the following message is written to the Gateway daemon error log:

```
CTG6507I Client disconnected: <connection details>,
 reason = CTG6656E Incorrect data <data> received on the network connection
 between the Gateway daemon and client application
```

#### Probable cause

The web service client application attempted to send a web service request to the Gateway daemon TCP, SSL, statistics API or local administration port, instead of the HTTP or HTTPS port.

## Action

Correct the web service client application to send web service requests to the Gateway daemon HTTP or HTTPS port.

# Chapter 101. 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 161 and “Client daemon logging” on page 184 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

On UNIX and Linux platforms, 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:

### UNIX and Linux:

| Platform | Server log location | Client log location |
|---|---|---|
| IBM 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.

### Windows:

| Product | Log location |
|---|---|
| IBM Communications Server API client | C:\IBMCS\w32cli\sna.err |
| IBM Communications Server | C:\IBMCS\PCWMSG.MLG |
| Microsoft Host Integration Server | Launch Windows Event Viewer and view the Application log |
| Microsoft Host Integration Server Client | Launch Windows Event Viewer and view the Application log |

For information about starting Windows Event Viewer see the documentation supplied with your operating system.

# Chapter 102. Tracing

Tracing can be enabled and controlled for different components of the CICS Transaction Gateway.

**Note:** Tracing, especially debug tracing, decreases performance.

**Related reference**:

Chapter 104, “Transaction tracking,” on page 515
You can use transaction tracking for Java, ECI V2, and .NET client requests or use Cross Component Trace for requests between WebSphere Application Server, CICS Transaction Gateway and CICS.

## 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 345.

### Tracing on Windows:

For information about enabling trace at startup using the configuration file see Chapter 47, “Configuring trace settings,” on page 233.

You can enable trace so that it starts whenever you start the Gateway daemon as a Windows service by configuring the Gateway daemon.

Use the **-A** option on the **ctgservice** command.

### Tracing on UNIX and Linux:

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 in console mode.

### 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 `/var/cicscli/gateway.trc` on UNIX and Linux and `<product_data_path>\gateway.trc` on Windows by default.

To override the default destination for trace output on UNIX and Linux, 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.

**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 350.

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 on UNIX and Linux or <product_data_path> subdirectory on Windows. 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.wrp***n*
The binary trace file generated when memory mapped tracing is enabled, where *n* is a number in the range 1 to 200.

**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 502 for information on the trace conversion utility.

## Wrapping the Client trace

Wrapping trace defines the amount of disk space used by the Client daemon binary trace files.

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 on, the binary trace files are created as part of Client daemon initialization.

When wrapping trace is turned on without memory mapped trace, two binary trace 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.

When wrapping trace is turned on with memory mapped trace, multiple binary trace files are used, for more details refer to "Memory mapped tracing."

## Memory mapped tracing

Memory mapped trace is the most efficient way to gather Client daemon trace. Memory mapped trace requires that wrapping trace is enabled.

With memory mapped tracing, the operating system's paging mechanism is used to swap trace data between memory and the wrapped trace files. 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 tracing to file instead of memory mapped tracing.

If you use memory mapped tracing, the size of the trace files is limited to 10 million bytes. 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” on page 235 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 10 million bytes of memory.

See “Starting client tracing” on page 350 for details of how to issue the command, and for UNIX and Linux see “Security considerations for UNIX and Linux systems” on page 353 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`.For more information, 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.

```
       -->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 31. 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 Part 11, “Performance,” on page 411 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 502 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 Color 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 Color Turquoise)
 Data : '9'
 Set Buffer Address to (24,51)
 Start Field Extended (Unprotected,Alphanumeric,Intense,pen-detectable,Foreground Color 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 32. 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.

## Tracing in ECI V2 and ESI V2 applications

Applications should implement an option to enable trace. You can control tracing in ECI and ESI Version 2 applications using the functions and environment variables described here.

You can set trace level, file, data length and offset either by using a function call or by setting an environment variable. Examples of each are shown below. To avoid having to recompile applications, enable trace by setting the environment variable.

### Trace level

You can set 5 trace levels:

**CTG_TRACE_LEVEL0**
Disables all tracing. This is the default setting.

**CTG_TRACE_LEVEL1**
Enables exception trace points. This level of tracing can be set on permanently to provide an error log capability. Messages are written only for system errors, socket errors, and other Gateway connection errors.

**CTG_TRACE_LEVEL2**
Enables event trace points and those from lower trace levels.

**CTG_TRACE_LEVEL3**
Enables function entry and exit trace points and those from lower trace levels.

**CTG_TRACE_LEVEL4**
Enables debug trace points and those from lower trace levels.

Here is an example of the trace level function call:

```
CTG_setAPITraceLevel(CTG_TRACE_LEVEL1);
```

Here is an example of the trace level environment variable:

```
CTG_CLIENT_TRACE_LEVEL=1
```

### Trace file

The default trace destination is the standard error stream.

Here is an example of the trace file function call:

```
CTG_setAPITraceFile("filename.trc");
```

Here is an example of the trace file environment variable:

```
CTG_CLIENT_TRACE_FILE=filename.trc
```

If the trace file is not set, trace is written to the standard error stream (stderr).

### Trace data length

The trace data length specifies the maximum amount of data that is written to trace when communicating with CICS Transaction Gateway and the trace level is set to CTG_TRACE_LEVEL4. The default setting is 128 bytes.

Here is an example of the trace data length function call:

```
CTG_setAPITraceDataLength(256);
```

Here is an example of the trace data length environment variable:

```
CTG_CLIENT_DATA_LENGTH=256
```

### Trace data offset

The trace data offset specifies an offset into data where tracing begins. When combined with the trace data length this allows a specific section of data to be traced, for example a section of data in a COMMAREA. The default setting is zero.

Here is an example of the trace data offset function call:

```
CTG_setAPITraceDataOffset(40);
```

Here is an example of the trace data offset environment variable:

```
CTG_CLIENT_DATA_OFFSET=40
```

## 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: Developing Applications* for further information.

### Tracing in Java Applets

When using Java Applets on Windows, data written to the error stream can be viewed using the Java Console. See your Java documentation for information on how to enable your Java Console.

## JNI tracing

JNI trace can be enabled on start up of CICS Transaction Gateway or a local mode application. Alternatively, JNI trace can be enabled when the Gateway daemon is running using the **ctgadmin** command.

Set the following environment variables before you start the CICS Transaction Gateway on UNIX and Linux or Java Client applications run in local mode on any platform:

•

**CTG_JNI_TRACE**
Use this environment variable to set the name of the JNI trace file. This environment variable defines only the name of the JNI trace file; it does not enable trace. There is no requirement to use a particular extension for the file name. If a directory is not defined the file is created in the current working directory.

**CTG_JNI_TRACE_ON**
Set this environment variable to YES (not case sensitive) to enable JNI trace. When **CTG_JNI_TRACE** is not defined and **CTG_JNI_TRACE_ON=YES**, trace is written to `stderr`.

Use one of the following methods to enable JNI trace:

- When you start the CICS Transaction Gateway on Windows, use the **ctgservice** command to register startup override options. For example:

  ```
  ctgservice -R -A-j-Dgateway.T.setJNITFile=filename
  ```

  This setting is defined in the Windows registry and persists. To remove the startup parameters issue the command, **ctgservice -R**. For more information, see the "ctgservice command reference" on page 365.
- When you start the CICS Transaction Gateway on UNIX and Linux, use the **ctgstart** command to register startup override options. For example:

  ```
  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 `/var/cicscli`.
- While the CICS Transaction Gateway is running, use the **ctgadmin** command. For example:

  ```
  ctgadmin -a trace -jnilevel=1 -jnifile=filename
  ```

  If a directory is not defined the file is created in the <product_data_path> directory. For more information, see "ctgadmin command reference" on page 361.
- For Java Client applications run 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.

## 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 |

| Level | Trace |
|---|---|
| 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.

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.

On Windows this might map to:

```
C:\Documents and Settings\Administrator\IBM\ctg\
```

On UNIX and Linux 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 nonmanaged 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.

# Tracing for Microsoft NET Framework-based client programs

Trace is activated for the `IBM.CTG.Client.dll` either by specifying it as an application configuration file or by using the Trace class.

## Trace levels

The following trace levels are available:

**CtgTrcDisabled**
disables tracing

**CtgTrcLevel1**
includes exception trace points but nothing else

**CtgTrcLevel2**
includes event trace points and all CtgTrcLevel1 trace points

**CtgTrcLevel3**
includes function entry and exit trace points and all CtgTrcLevel1 and CtgTrcLevel2 trace points

**CtgTrcLevel4**
includes debug trace points and all CtgTrcLevel1, CtgTrcLevel2 and CtgTrcLevel3 trace points (the most verbose tracing level)

## Specifying trace in an application configuration file

Trace can be enabled using the CtgTrace trace switch in an application configuration file (an XML file). The switch allows the trace to be specified as an IBM.CTG.TraceLevel value, a System.Diagnostics.TraceLevel value, or an integer between 0 and 4 inclusive. In the following example the switch `value="CtgTrcLevel4"` specifies Level 4 tracing, with tracing of data blocks limited to the first 128 bytes.

```
<?xml version="1.0" encoding="utf-8" ?>
 <configuration>
  <system.diagnostics>
   <switches>
    <add name="CtgTrace" value="CtgTrcLevel4" dataDumpOffset="0"
dataDumpLength="128"/>
   </switches>
  </system.diagnostics>
 </configuration>
```

A sample trace configuration file called `App.config` is included in the SDK package or in `<install_path>\samples\csharp\eci` and `<install_path>\samples\vb\eci` on a Windows machine with CICS Transaction Gateway installed.

## Using the Trace class

The Trace class includes the following members:

**TraceLevel**
gets or sets the trace level

**DataDumpOffset**
gets or sets the starting offset in each data blocks when tracing at CtgTrcLevel4

**DataDumpLength**
gets or sets the maximum amount of data traced in each data block at CtgTrcLevel4

For more information see the Trace information in the .NET section of the *Programming Reference*.

# Chapter 103. 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 documentation.
2. Search the Internet. If you cannot find an answer to your question in the documentation, 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.

# Part 15. 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.

# Chapter 104. Transaction tracking

You can use transaction tracking for Java, ECI V2, and .NET client requests or use Cross Component Trace for requests between WebSphere Application Server, CICS Transaction Gateway and CICS.

## Transaction tracking across an IBM CICSPlex

Transaction tracking can assist in diagnosing problems that sometimes occur when complex distributed transactions spread across a IBM CICSPlex.

Transaction tracking is available for Java, ECI V2 and NET Framework-based client requests, through the availability of origin data to the monitoring exits. Origin data associated with each transaction is forwarded by CICS on each subsequent DPL between CICS regions. This enables tracking of requests associated with a given client application, as they pass through the Gateway daemon, through the connected CICS servers, to the target programs in CICS.

Any tasks in CICS initiated using ECI through IPIC connections have associated origin data in CICS carrying a fully qualified APPLID field. Origin data in CICS can be viewed using IBM CICSPlex SM or using the **INQUIRE ASSOCIATION** SPI command. You can also use the CICS Transaction Gateway request monitoring exits in the Gateway daemon to view the origin data, and Java client applications have access to origin data in their request monitoring exits. These commands allow an administrator to identify the Client application that originated a particular task.

If you specify an APPLID and APPLID qualifier for the Client application, they are used in the origin data. If they are not specified, but you are running in remote mode and the values are specified in the configuration file, these values are used. If an APPLID and APPLID qualifier are not specified at all, the values automatically generated by CICS for the IPIC connection are used. The fully qualified APPLID can be viewed in CICS using the **CEMT INQUIRE IPCONN** command. It is displayed in the *Applid* and *NetworkId* fields.

**Related concepts**:

"Client APPLID and APPLID qualifier" on page 108
Set the APPLID and APPLID qualifier for Client applications to enable transaction tracking.

**Related reference**:

"Gateway APPLID" on page 107
The **APPLID** parameter identifies the instance of the CICS Transaction Gateway on server connections and tasks in a CICSplex.

## Transaction tracking with Cross Component Trace (XCT)

You can use Cross Component Trace (XCT) to track individual requests as they flow between IBM WebSphere Application Server, CICS Transaction Gateway and CICS, assisting both problem diagnosis and system planning and configuration.

The XCT facility is available when using IPIC connections and IBM WebSphere Application Server V8.5 or later, when High Performance Extensible Logging (HPEL) is enabled. As a request flows through the system, the related XCT information is in the IBM WebSphere HPEL log, CICS TG request monitoring exit

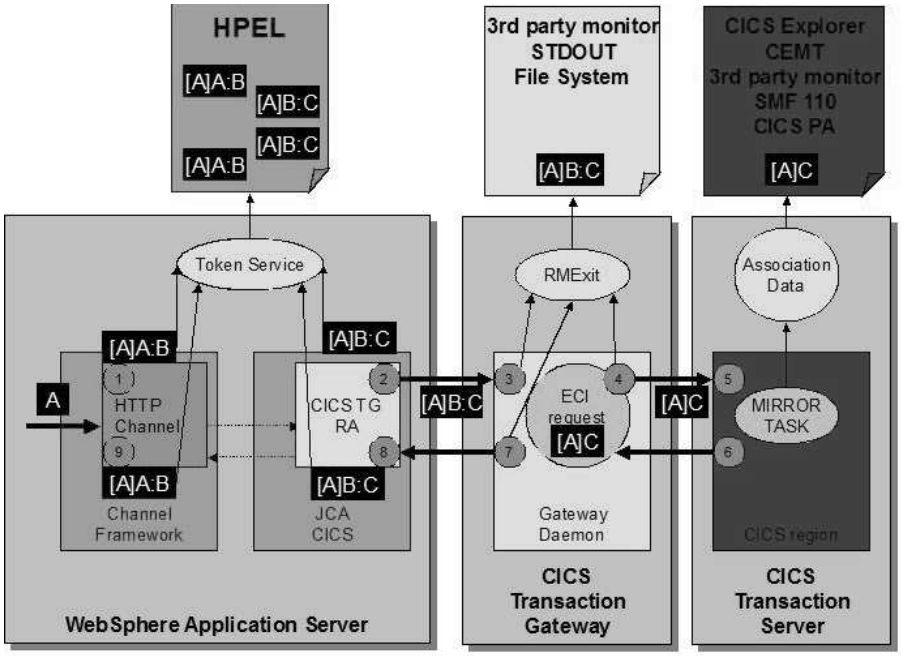
data and CICS task association data, which you can view in a variety of ways.

## XCT contexts

XCT contexts are hierarchical and Begin and End demarcate component boundaries.

A thread of execution can have up to three XCT contexts at any one time:

- Root – the initial context (Request ID) of the component at the point of entry.
- Parent – the context of the calling component.
- Current – the context of the current component.



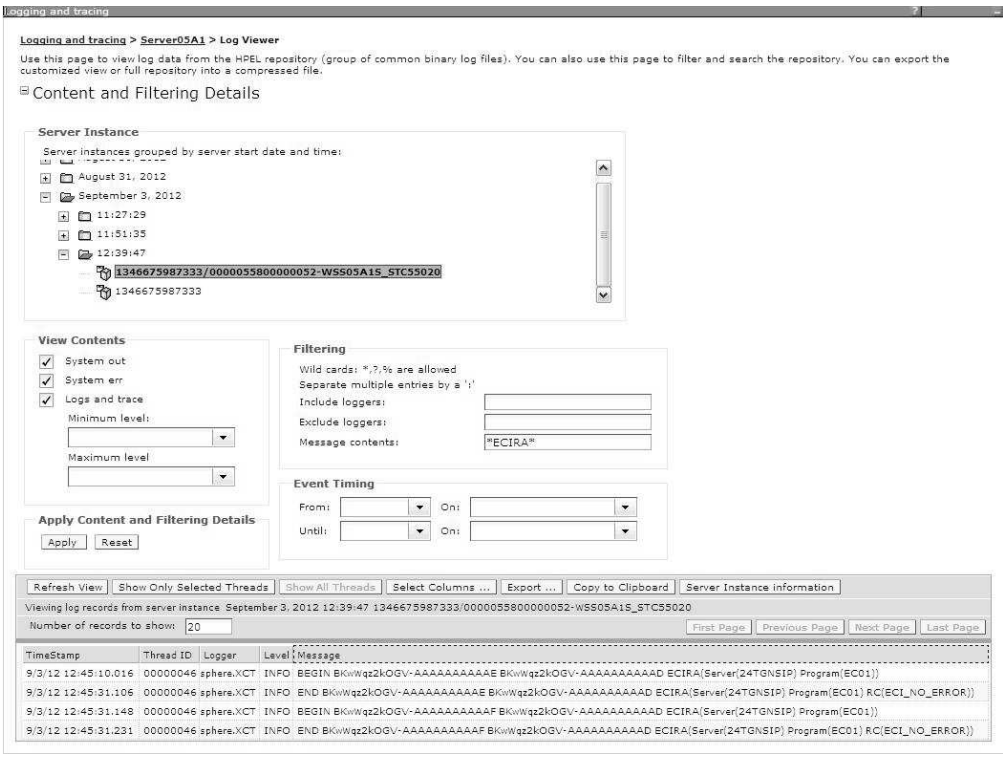

In the diagram above, A refers to the context assigned to the incoming request, B to the context of the channel framework, and C to the JCA CICS.

## XCT data in IBM WebSphere

XCT context data is written to the HPEL log.

For the ECI resource adapter, the log is annotated with the target CICS server, the CICS program name and, on exit, the CICS return code.

An example of XCT context data written to the HPEL repository viewed through the IBM WebSphere Application Server Log Viewer:




The XCT context data can also be seen in the IBM WebSphere Application Server log file:

```
[9/3/12 12:45:10:016 GMT] 00000046  I UOW= source=com.ibm.websphere.XCT class= method= org=null prod=null component=null
          thread=[WebSphere WLM Dispatch Thread t=007c40b8] requestID=[BKwWqX+HPuK-AAAAAAAAAAG]
          BEGIN BKwWqz2kOGV-AAAAAAAAAAE BKwWqz2kOGV-AAAAAAAAAAD ECIRA(Server(24TGNSIP) Program(EC01))
[9/3/12 12:45:31:106 GMT] 00000046  I UOW= source=com.ibm.websphere.XCT class= method= org=null prod=null component=null
          thread=[WebSphere WLM Dispatch Thread t=007c40b8] requestID=[BKwWqX+HPuK-AAAAAAAAAAG]
          END   BKwWqz2kOGV-AAAAAAAAAAE BKwWqz2kOGV-AAAAAAAAAAD ECIRA(Server(24TGNSIP) Program(EC01) RC(ECI_NO_ERROR))
```

Additional information can be viewed using the IBM WebSphere Application Server Cross Component Trace.

For more information about configuring Cross Component Trace, see the IBM WebSphere Application Server V8.5 documentation.

## XCT data in CICS TG

Within the Gateway daemon, the XCT context data is available in the request monitoring exits.

At all request monitoring exit points, three XCT request identifiers are available:

- XctRoot, the initial context of the component at the point of entry.
- XctParent, the context of the calling component.
- XctCurrent, the context of the current component, CICS TG.

The XCT data in the Gateway daemon is available for all CICS server protocols but it is only when using IPIC that the context data is propagated to CICS in the origin data. The root and current XCT contexts are available in the origin data at the RequestDetails and ResponseExit exit points.

The following example shows the XCT and origin data entries in the request monitoring log when using the supplied sample request monitor BasicMonitor:

```
[00000000001]: com.ibm.ctg.samples.requestexit.BasicMonitor:eventFired called with event = RequestEntry
  : : : : : :
[00000000001]: XctRoot = BKwWqX+HPuK-AAAAAAAAAG
[00000000001]: XctParent = BKwWqz2kOGV-AAAAAAAAAD
[00000000001]: XctCurrent = BKwWqz2kOGV-AAAAAAAAAE

[00000000001]: com.ibm.ctg.samples.requestexit.BasicMonitor:eventFired called with event = RequestDetails
 : : : : : :
[00000000001]: OriginData - Transaction Group ID = 1A10C2C1 E8D3C9E2 E22EC7C1 E3C5E6C1 E8F1CA1D BCBD6459 8200
                          - User Correlator = XCT     BKwWqX+HPuK-AAAAAAAAAG BKwWqz2kOGV-AAAAAAAAAE
 : : : : : :
[00000000001]: XctRoot = BKwWqX+HPuK-AAAAAAAAAG
[00000000001]: XctParent = BKwWqz2kOGV-AAAAAAAAAD
[00000000001]: XctCurrent = BKwWqz2kOGV-AAAAAAAAAE

[00000000001]: com.ibm.ctg.samples.requestexit.BasicMonitor:eventFired called with event = ResponseExit
 : : : : : :
[00000000001]: OriginData - Transaction Group ID = 1A10C2C1 E8D3C9E2 E22EC7C1 E3C5E6C1 E8F1CA1D BCBD6459 8200
                          - User Correlator = XCT     BKwWqX+HPuK-AAAAAAAAAG BKwWqz2kOGV-AAAAAAAAAE
 : : : : : :
[00000000001]: XctRoot = BKwWqX+HPuK-AAAAAAAAAG
[00000000001]: XctParent = BKwWqz2kOGV-AAAAAAAAAD
[00000000001]: XctCurrent = BKwWqz2kOGV-AAAAAAAAAE
```

For more information about when the XCT request identifiers are available, see *Data available by FlowType and RequestEvent* in *CICS TG Developing Applications*.

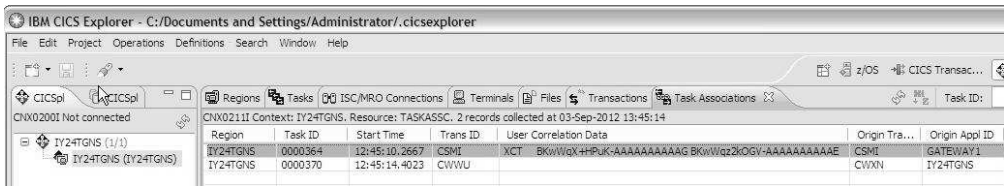
## XCT data in CICS

For IPIC only, the user correlation data containing the root and current XCT contexts is sent from CICS TG to CICS as part of the origin data.

In addition to obtaining the user correlation data through the CICS API, the user correlation data can be viewed in the task's association data, using the command CEMT INQUIRE ASSOCIATION(taskid), or in the Task Associations view in CICS Explorer, and is recorded to SMF in type 110, sub-type 01 records.

The following screenshot shows association data displayed using the CICS API command INQUIRE ASSOCIATION(taskid):


The CICS Explorer can also be used to display the user correlation data, for example:



# Chapter 105. 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 31. Data available to request monitoring exits*

| Request data description | Description |
|---|---|
| Channel | Channel that is associated with the request. |
| CicsAbendCode | CICS abend code on a response. |
| CicsReturnCode | CICS return code on a response. |
| CicsServer | Server to which CICS Transaction Gateway sent the request. |
| ClientCtgApplid | APPLID of the Client application. |
| ClientCtgApplidQualifier | APPLID qualifier of the Client application. |
| ClientCtgCorrelator | Correlator that was generated by the Java client application. |
| ClientLocation | The location of the Client application (IP address). |
| ClientProtocol | Protocol that was used by the client application to send the request. |
| ClientType | Type of the client application that sent the request. |
| ClientVersion | Version of the CICS TG client that sent the request. |
| CommandData | Command data that originated from a request monitor exit administration request. |
| CtgApplid | If the FlowTopology is "Gateway", this is the Gateway daemon APPLID, otherwise it is the client application APPLID. |
| CtgApplidQualifier | If the FlowTopology is "Gateway" this is the Gateway daemon APPLID qualifier, otherwise it is the client application APPLID qualifier. |
| CtgCorrelator | CICS Transaction Gateway identifier that is used to track this flow within the CICS Transaction Gateway instance. |
| CtgReturnCode | CICS Transaction Gateway return code on a response. |
| DistributedIdentity | Distributed identity that is associated with this transaction. |
| FlowTopology | Topology from which the request exit was called:<br>• Gateway - from the Gateway daemon<br>• RemoteClient - from a remote client<br>• LocalClient - from a local client |

*Table 31. Data available to request monitoring exits (continued)*

| Request data description | Description |
|---|---|
| FlowType | Flow type of this request or response. |
| GatewayUrl | URL of the Gateway to which the Java client is connecting. |
| HttpPath | URI that the request was sent to. |
| HttpPayload | HTTP payload data (JSON) that is associated with the request or the response. |
| HttpStatusCode | HTTP status code on a response. |
| HttpVerb | HTTP verb for this request. |
| Location | Location of this monitor. The value is an IP address. |
| LuwToken | CICS Transaction Gateway logical unit of work token. |
| OriginData | Data identifying the Client application that originated a CICS task and that contains the APPLID and APPLID qualifier. This data is available only when using the IPIC protocol. |
| PayLoad | 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 request 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 | 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 remote client. |
| WorkerWaitTime | Period 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. |
| XctCurrent | The identifier for this transaction. |
| XctParent | The identifier for the parent of this transaction. |
| XctRoot | The root identifier for this transaction. |
| Xid | XID for XA transaction. |

**Related concepts**:

"Request monitoring exits configuration" on page 521
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 can only configure request monitoring exits for the Gateway classes.

**Related information**:

Java request monitoring exits

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 can only 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 request monitoring exits for the Gateway daemon" on page 219.

# Chapter 106. 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. During a normal shutdown of the CICS Transaction Gateway the statistics are written to the Gateway daemon information log.

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 section of the *CICS Transaction Server for IBM z/OS Information Center*.

## Resource group ID

Statistics resource groups are a logical grouping of resources such as connection manager threads, and represented by resource group IDs. 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 533 for a list statistic IDs arranged by resource group.

## Statistical type

There are four statistical types: C (Current), I (Interval), L (Lifetime) and S (Startup). For more information, see “List of statistics” on page 533.

### 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.

### Statistics recording to a log file

CICS Transaction Gateway provides a mechanism to record interval, end-of-day data, and shutdown statistics data to a statistics log file. For more information, see “Recording statistics to a file” on page 550.

## 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 526.

### 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” on page 222 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.
- 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” on page 222 for more information.

### Recording to a file

CICS Transaction Gateway provides a mechanism for recording interval statistics and end of day data to a file. You can set the **statsrecording** and **log@statistics** parameters using the Gateway settings in the configuration file. For more information, see Configuring statistics recording to a file.

### 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.
- The Gateway section of the configuration file.

See "Statistics API protocol settings" on page 220 for more information.

**Related concepts**:

"Recording statistics to a file" on page 550
You can record statistics data from CICS Transaction Gateway to an XML file at predefined intervals.

**Related reference**:

"Statistics interval" on page 222
The **statint** parameter specifies the recording interval for system statistics.

"Statistics end of day time" on page 222
The **stateod** parameter specifies the end of day time.

"Statistics API protocol settings" on page 220
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**:

"GATEWAY section of the configuration file" on page 240
The GATEWAY section of the configuration file defines the Gateway daemon settings.

## 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.

### Procedure

1. Navigate to the **Statistics API** settings page.
2. Select the **Enable protocol handler** check box.
3. Optional: enter a value in the **Port** field if the default setting is not suitable.
4. Navigate to the **Monitoring** tab of the Gateway daemon node.
5. Enter a value for the **Statistics interval** in the format `HHMMSS`. The default value is three hours (`030000`).
6. Enter a value for the **Statistics End of Day time** in the format `HHMMSS`. The default value is midnight (`000000`).

7. Select the **Enable statistics recording** check box.
8. Enter a value for the XML statistics log file in the **Statistics log filename**.
9. Save the configuration file, then stop and restart the Gateway daemon.

**Related reference**:

"Statistics API protocol settings" on page 220
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 243
To enable the statistics API protocol, include a protocol handler definition in the GATEWAY section of the configuration file.

## Interval statistics

The **statint** keyword shows the statistics interval duration and the **stateod** keyword 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`:

```
# StatInt = 030000 # Statistics interval in the form HHMMSS
# StatEOD = 000000 # Statistics end of day time in the form HHMMSS
```

The two fields **Statistics interval** and **Statistics End of Day time** are shown in the Monitoring tab of the Gateway daemon panel in the Configuration Tool.

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.

The **statint** and **stateod** keywords are in the "GATEWAY section of the configuration file" on page 240.

## 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 32. 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 33. 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 |

*Table 33. Interval Statistics timing – Example 2 (continued)*

| Time | Event type | Interval length HH:MM:SS |
|---|---|---|
| 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 34. 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 the **ctgadmin** command, with the **-si** parameter followed by a list of resource group 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.

Issue the following command:

```
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 35. 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 configured name of the CICS server. |

*Table 35. Resource groups (continued)*

| ID | Name | Description |
|---|---|---|
| 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. |
| WS | Web service (all) statistics | Statistics about all web services. |
| WS*x* | Web service (instance) statistics | Statistics for an individual web service, where *x* is the name of the web service. |
| 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.

CICS server statistics will be taken for the CICS servers that are actually used to process the work. When the Workload Manager is in use, the CICS server identified by an application might not be the CICS server that processes the work.

## 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.

The statistics in the GD resource group are not updated by web service requests. Statistics for web service requests are provided by the WS resource group.

## Protocol handler resource group (PH)

Statistics are available to identify whether the TCP, SSL, HTTP, and HTTPS protocol handlers are enabled and on which address and port they are accepting requests. A port value of -1 is returned if the protocol is not enabled.

## 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.

## Web service (all) resource group (WS)

Statistics are available that summarize interactions with all associated web services.

## Web service (instance) resource group (WSx)

Statistics are available for each specific web service *x*. In general, the statistic IDs of the WS resource group which summarize activity across all associated web services, can be found for each web service in the corresponding WSx resource group.

For example, a request to a web service defined by the name EC03 is represented by resource group WSEC03. An example of a statistical ID available for such a resource group is WSEC03_LALLREQ.

## Worker thread resource group (WT)

Statistics are available for the worker threads. These statistics identify the characteristics for the pool of worker threads. 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.

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.

**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.

**Note:** Web services work is not included in these statistics.

*Table 36. 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<br>**Note:** A value of -1 indicates no limit. |

## CICS server (all) statistics

The statistics listed here belong to the CICS server (all) resource group.

**Note:** Web services work is included in these statistics. The CICS Server(CS) statistics are only incremented in the case where the web server passes the request

to the CICS server. If the request is rejected by the web service, the web service(WS) statistics are incremented, but the CICS Server statistics are not.

*Table 37. 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_ICOUNT | The number of CICS servers to which requests have been sent. This number equals the number of CICS servers in CS_ILIST. | 0 | Integer |
| CS_ILIST | The list of CICS servers to which requests have been sent. | <empty string> | String |
| 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 connected CICS servers 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 |

*Table 37. CICS server (all) statistics (continued)*

| ID | Description | Default value | Data type |
|---|---|---|---|
| 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 |
| 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.

**Note:** Web services work is included in these statistics.

*Table 38. 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 |

*Table 38. CICS server (instance) statistics (continued)*

| ID | Description | Default value | Data type |
|---|---|---|---|
| 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*_CSTATUS | The status of the IPIC connection to CICS server x. Possible values are NOTSTARTED, STARTING, AVAILABLE, UNAVAILABLE, STOPPING, STOPPED.<br><br>For more information, see "CS*x*_CSTATUS values" on page 538. | NOTSTARTED | String |
| 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 |
| 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 |

*Table 38. CICS server (instance) statistics (continued)*

| ID | Description | Default value | Data type |
|---|---|---|---|
| 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, IPICSSL, SNA, TCPIP. | N/A | String |
| CS*x*_SSESSMAX | The number of requested IPIC sessions for CICS server *x*. | 0 | Integer |

**CS*x*_CSTATUS values:**

This table describes the possible values for CS*x*_CSTATUS.

*Table 39. CSx_CSTATUS values*

| CS*x*_CSTATUS value | Description | Event | New value |
|---|---|---|---|
| NOTSTARTED | Initial state of the server connection.<br><br>The Gateway accepts requests. | Start the server: `ctgadmin -a server -start=<IPIC Server>` | AVAILABLE if the connection is successful<br><br>UNAVAILABLE if the connection is unsuccessful |
| | | ECI or ESI request sent to the server. | AVAILABLE if the connection is successful<br><br>UNAVAILABLE if the connection is unsuccessful |
| STARTING | The server connection is starting | | AVAILABLE if the connection is successful<br><br>UNAVAILABLE if the connection is unsuccessful |
| AVAILABLE | The server is connected, available and the Gateway accepts requests. | Server idle timeout is reached. The `srvidletimeout` value is set for the server and has been reached with no outstanding work. | NOTSTARTED |

*Table 39. CSx_CSTATUS values (continued)*

| CS*x*_CSTATUS value | Description | Event | New value |
|---|---|---|---|
| | | Stop the server: `ctgadmin -a server -stop=<IPIC Server>` | STOPPING |
| | | Server becomes unavailable. For example, the server heartbeat fails at next heartbeat interval. | UNAVAILABLE |
| UNAVAILABLE | The connection has failed and the Gateway rejects requests. | Stop the server: `ctgadmin -a server -stop=<IPIC Server>` | STOPPING |
| | | The server becomes available after a successful retry. For example, after server retry interval is reached. | AVAILABLE |
| STOPPING | The connection is closing.<br><br>The Gateway rejects new requests.<br><br>The Gateway accepts requests that continue or complete outstanding work. | All outstanding work that used this connection is complete, or has been cancelled by a stop immediate command:<br><br>`ctgadmin -a server -stop=<IPIC Server> -immediate`<br>or<br>`ctgadmin -a server -stop=<IPIC Server> -imm` | STOPPED |
| | | Start the server: `ctgadmin -a server -start=<IPIC Server>` | AVAILABLE if the connection is successful<br><br>UNAVAILABLE if the connection is unsuccessful |
| STOPPED | The connection is closed and the Gateway rejects requests. | Start the server: `ctgadmin -a server -start=<IPIC Server>` | AVAILABLE if the connection is successful<br><br>UNAVAILABLE if the connection is unsuccessful |

## Web service (all) statistics

The statistics listed here belong to the web service (all) resource group.

*Table 40. Web service (all) statistics*

| ID | Description | Default value | Data type |
|---|---|---|---|
| WS_SCOUNT | The number of web services defined in the configuration file. | 0 | Integer |
| WS_SLIST | The list of all web services defined in the configuration file. | <empty string> | String |
| WS_SMAXHTTP | The maximum number of HTTP client threads for processing web service requests. | 0 | Integer |

*Table 40. Web service (all) statistics (continued)*

| ID | Description | Default value | Data type |
|---|---|---|---|
| WS_IALLREQ | The number of web service requests that have been processed. Successful and failed requests are included. | 0 | Integer |
| WS_LALLREQ | The number of web service requests that have been processed. Successful and failed requests are included. | 0 | Integer |
| WS_IAVRESP | The average time in milliseconds for web services to respond to 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 |
| WS_LAVRESP | The average time in milliseconds for web services to respond to 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 |
| WS_IREQDATA | The amount of application request data (in bytes) received from clients of all web services. | 0 | Integer |
| WS_LREQDATA | The amount of application request data (in bytes) received from clients of all web services. | 0 | Integer |
| WS_IRESPDATA | The amount of application response data (in bytes) returned to clients of all web services. | 0 | Integer |
| WS_LRESPDATA | The amount of application response data (in bytes) returned to clients of all web services. | 0 | Integer |
| WS_IREQHI | The peak number of concurrent web service requests that have been in flight at the same time. | 0 | Integer |
| WS_LREQHI | The peak number of concurrent web service requests that have been in flight at the same time. | 0 | Integer |
| WS_CREQ | The current number of inflight web service requests. These requests might or might not be active in CICS; however, they represent one mirror transaction, which might be in a suspended state. | 0 | Integer |
| WS_CWAITING | The current number of web service requests waiting for a worker thread to become available. | 0 | Integer |

## Web service (instance) statistics

The statistics listed here belong to the web service (instance) resource group.

*Table 41. Web service (instance) statistics*

| ID | Description | Default value | Data type |
|---|---|---|---|
| WSx_SSERVER | The CICS server that requests to this web service are sent. | None | String |

*Table 41. Web service (instance) statistics (continued)*

| ID | Description | Default value | Data type |
|---|---|---|---|
| WSx_SURI | The URI that requests to this web service are sent. | None | String |
| WSx_SEIBTRNID | The CICS EIBTRNID transaction identifier for requests to this web service. | None | String |
| WSx_SMIRROR | The name of the CICS mirror transaction under which requests to this web service run. | CSMI/CPMI | String |
| WSx_SPROGRAM | The name of the CICS program that implements this web service. | None | String |
| WSx_IALLREQ | The number of web service requests that have been processed. Successful and failed requests are included. | 0 | Integer |
| WSx_LALLREQ | The number of web service requests that have been processed. Successful and failed requests are included. | 0 | Integer |
| WSx_IAVRESP | The average time in milliseconds for web service x to respond to 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 |
| WSx_LAVRESP | The average time in milliseconds for web service x to respond to 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 |
| WSx_IREQDATA | The amount of application request data (in bytes) received from clients of web service x. | 0 | Integer |
| WSx_LREQDATA | The amount of application request data (in bytes) received from clients of web service x. | 0 | Integer |
| WSx_IRESPDATA | The amount of application response data (in bytes) returned to clients of web service x. | 0 | Integer |
| WSx_LRESPDATA | The amount of application response data (in bytes) returned to clients of web service x. | 0 | Integer |
| WSx_IREQHI | The peak number of concurrent web service transactions that have been in flight at the same time. | 0 | Integer |
| WSx_LREQHI | The peak number of concurrent web service transactions that have been in flight at the same time. | 0 | Integer |
| WSx_CREQ | The current number of inflight web service transactions. These transactions might or might not be active in CICS; however, they represent one mirror transaction, which might be in a suspended state. | 0 | Integer |

## Gateway daemon statistics

The statistics listed here are members of the Gateway daemon resource group.

**Note:** Web services work is not included in these statistics except for GD_LHAEXIT and GD_IHAEXIT.

*Table 42. 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) 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) 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, ESI, EPI) 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 |
| 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) requests that have been processed. Successful and failed requests are included. Administrative requests and handshakes are excluded. | 0 | Integer |

*Table 42. Gateway daemon statistics (continued)*

| ID | Description | Default value | Data type |
|---|---|---|---|
| GD_LAVRESP | The average time in milliseconds for the Gateway daemon to respond to API (ECI, ESI, EPI) 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, ESI, EPI) 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 |
| 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: IBM AIX, HP-UX (Itanium), Linux (Intel), Linux (POWER), Linux (IBM zSeries), Solaris, Windows, IBM 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 |

*Table 42. Gateway daemon statistics (continued)*

| ID | Description | Default value | Data type |
|---|---|---|---|
| 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 43. 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) and web service 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 44. 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 |

*Table 44. System Environment statistics (continued)*

| ID | Description | Default value | Data type |
|---|---|---|---|
| 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 |
| 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 45. Protocol handler statistics*

| ID | Description | Default value | Data type |
|---|---|---|---|
| PH_SBINDHTTP | The address or host name to which the HTTP protocol handler is bound. This statistic does not contain a value if the protocol handler is not enabled. | none | String |
| PH_SBINDHTTPS | The address or host name to which the HTTPS protocol handler is bound. This statistic does not contain a value if the protocol handler is not enabled. | none | String |
| 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_SPORTHTTP | The HTTP protocol handler port number, or -1 if the protocol is not enabled. | -1 | Integer |
| PH_SPORTHTTPS | The HTTPS protocol handler port number, or -1 if the protocol is not enabled. | -1 | Integer |
| 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.

**Note:** All these statistics include web services work.

*Table 46. Worker thread statistics*

| ID | Description | Default value | Data type |
|---|---|---|---|
| WT_CALLOC | The current number of worker threads that are processing requests. | 0 | Integer |
| WT_CCURR | The current number of worker threads created. | 0 | Integer |
| WT_IALLOCHI | The peak number of worker threads concurrently in use. This number represents a high water mark for WT_CALLOC. | <WT_CALLOC> | Integer |
| WT_ITIMEOUTS | The number of times a worker thread could not be allocated within the defined **workertimeout** length of time. | 0 | Integer |
| WT_LTIMEOUTS | The number of times a worker thread could not be allocated 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<br>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 ICOUNT**
The number of CICS servers to which requests have been sent. This number equals the number of CICS servers in CS_ILIST.

**CS_ILIST**
The list of CICS servers to which requests have been sent.

**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.

## Recording statistics to a file

You can record statistics data from CICS Transaction Gateway to an XML file at predefined intervals.

For more information about how to configure recording statistics to a XML file, see "Configuring statistics recording to a file" on page 223.

### The XML statistics log file

The XML statistics log file stores historical data about your CICS Transaction Gateway.

You can use the data stored in the statistics log file to analyze the performance, diagnose system problems, monitor system resources, and determine transaction throughput of your CICS Transaction Gateway. You can analyze the statistic log file by viewing the raw XML data, or you can use a program application to analyze the information for you. If you use a program application, you can use the XML schema to validate that the XML file is well formed. The program application can also use the schema to parse the information in the log file. The XML schema file, `ctgstatslog.xsd`, is installed in the `<install_path>/docs` directory. For more information about using statistics, see "Using the statistics" on page 546.

# Chapter 107. 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.

# Part 16. 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.

# Chapter 108. Data conversion on Windows

The ASCII CCSID is determined dynamically. By default the GetOEMCP function is used to obtain the current OEM code page identifier, for example CCSID 850, for the system.

If the **Use OEM code page** configuration setting is specified, the GetACP function is used to obtain the current ANSI code page identifier, for example CCSID 1252, for the system.

The identifier might not be unique. For example, there are differences between IBM CCSID 932 and Microsoft CCSID 932 - the Client daemon maps the Microsoft one to CCSID 943 which is the equivalent one defined in the CRDA Registry. The Microsoft CCSID 1252 might be the pre-euro or post-euro version - the post-euro is mapped to CCSID 5348.

# Chapter 109. 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 IBM zSeries*, 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 47. 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 |
| 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 IBM z Systems products that support the CCSID. Three levels of support are defined: “Base”, “T01”, and “T02”.

  **Base**
  - CICS Transaction Server for IBM z/OS
  - CICS TS for VSE

  **T01**
  - CICS Transaction Server for IBM z/OS
  - CICS TS for VSE

  **T02**
  - CICS Transaction Server for IBM z/OS

## Arabic

A list of the Arabic client and server CCSIDs.

*Table 48. 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 49. 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 50. 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 51. 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 52. 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 53. 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 54. 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 55. 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 56. 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 57. 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 58. 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 |
| 1255 | T01 | 01255 | 01255 | MS Windows: Hebrew |
| 5351 | T02 | 05351 | 05351 | MS Windows: Hebrew, version 2 with euro |

*Table 59. 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 60. Japanese ASCII*

| CLINTCP | in | CCSID | CPGID | Comments |
|---|---|---|---|---|
| 932 | Base | 00932 | 1. 00897<br>2. 00301 | 1. PC data: SBCS<br>2. PC data: DBCS including 1880 user-defined characters |
| 942 | Base | 00942 | 1. 01041<br>2. 00301 | 1. PC data: Extended SBCS<br>2. PC data: DBCS including 1880 user-defined characters |
| 943 | T01 | 00943 | 1. 00897<br>2. 00941 | 1. PC data: SBCS<br>2. PC data: DBCS for Open environment including 1880 IBM user-defined characters |
| 954<br>EUCJP | Base | 00954 | 1. 00895<br>2. 00952<br>3. 00896<br>4. 00953 | 1. G0: JIS X201 Roman<br>2. G1: JIS X208-1990<br>3. G1: JIS X201 Katakana<br>4. G1: JIS X212 |
| 5050 | T02 | 05050 | 1. 00895<br>2. 00952<br>3. 00896<br>4. 00953 | 1. G0: JIS X201 Roman<br>2. G1: JIS X208-1990<br>3. G1: JIS X201 Katakana<br>4. G1: JIS X212 |

*Table 61. Japanese EBCDIC*

| SRVERCP | | CCSID | CPGID | Comments |
|---|---|---|---|---|
| 930 | Base | 00930 | 1. 00290<br>2. 00300<br>3. 00290<br>4. 00300 | 1. Katakana Host: extended SBCS<br>2. Kanji Host: DBCS including 4370 user-defined characters<br>3. Katakana Host: extended SBCS<br>4. Kanji Host: DBCS including 1880 user-defined characters |
| 931 | Base | 00931 | 1. 00037<br>2. 00300 | 1. Latin Host: SBCS<br>2. Kanji Host: DBCS including 4370 user-defined characters |
| 939 | Base | 00939 | 1. 01027<br>2. 00300<br>3. 01027<br>4. 00300 | 1. Latin Host: extended SBCS<br>2. Kanji Host: DBCS including 4370 user-defined characters<br>3. Latin Host: extended SBCS<br>4. Kanji Host: DBCS including 1880 user-defined characters |
| 1390 | T02 | 01390 | 1. 00290<br>2. 00300 | 1. Katakana Host: extended SBCS; with euro<br>2. Kanji Host: DBCS including 6205 user-defined characters |
| 1399 | T02 | 01399 | 1. 01027<br>2. 00300 | 1. Latin Host: extended SBCS; with euro<br>2. Kanji Host: DBCS including 4370 user-defined characters; with euro |

## Korean

A list of the Korean ASCII and EBCDIC.

*Table 62. Korean ASCII*

| CLINTCP | in | CCSID | CPGID | Comments |
|---|---|---|---|---|
| 934 | Base | 00934 | 1. 00891<br>2. 00926 | 1. PC data: SBCS<br>2. PC data: DBCS including 1880 user-defined characters |
| 944 | Base | 00944 | 1. 01040<br>2. 00926 | 1. PC data: Extended SBCS<br>2. PC data: DBCS including 1880 user-defined characters |
| 949 | Base | 00949 | 1. 01088<br>2. 00951 | 1. IBM KS Code - PC data: SBCS<br>2. IBM KS code - PC data: DBCS including 1880 user-defined characters |
| 970 EUCKR | Base | 00970 | 1. 00367<br>2. 00971 | 1. G0: ASCII<br>2. G1: KSC X5601-1989 including 1880 user-defined characters |
| 1363 | T01 | 01363 | 1. 01126<br>2. 01362 | 1. PC data: MS Windows Korean SBCS<br>2. PC data: MS Windows Koran DBCS including 11172 full Hangul |

*Table 63. Korean EBCDIC*

| SRVERCP | in | CCSID | CPGID | Comments |
|---|---|---|---|---|
| 933 | Base | 00933 | 1. 00833<br>2. 00834 | 1. Host: Extended SBCS<br>2. Host: DBCS including 1880 user-defined characters and 11172 full Hangul characters |
| 1364 | T02 | 01364 | 1. 00833<br>2. 00834 | 1. Host: Extended SBCS<br>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 64. 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 65. 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 66. Latin-2 ASCII*

| CLINTCP | in | CCSID | CPGID | Comments |
|---|---|---|---|---|
| 852 | Base | 00852 | 00852 | PC data: Latin-2 multilingual |

*Table 66. Latin-2 ASCII (continued)*

| CLINTCP | in | CCSID | CPGID | Comments |
|---|---|---|---|---|
| 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 67. 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 68. 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 69. 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 70. Simplified Chinese ASCII*

| CLINTCP | in | CCSID | CPGID | Comments |
|---|---|---|---|---|
| 946 | Base | 00946 | 1. 01042<br>2. 00928 | 1. PC data: Extended SBCS<br>2. PC data: DBCS including 1880 user-defined characters |
| 1381 | Base | 01381 | 1. 01115<br>2. 01380 | 1. PC data: Extended SBCS (IBM GB)<br>2. PC data: DBCS (IBM GB) including 31 IBM-selected, 1880 user-defined characters |

*Table 70. Simplified Chinese ASCII (continued)*

| CLINTCP | in | CCSID | CPGID | Comments |
|---|---|---|---|---|
| 1383<br>(EUCCN) | T01 | 01383 | 1. 00367<br>2. 01382 | 1. G0: ASCII<br>2. G1: GB 2312-80 set |
| 1386 | T01 | 01386 | 1. 01114<br>2. 01385 | 1. PC data: S-Chinese GBK and T-Chinese IBM BIG-5<br>2. PC data: S-Chinese GBK |

*Table 71. Simplified Chinese EBCDIC*

| SRVERCP | in | CCSID | CPGID | Comments |
|---|---|---|---|---|
| 935 | Base | 00935 | 1. 00836<br>2. 00837 | 1. Host: Extended SBCS<br>2. Host: DBCS including 1880 user-defined characters |
| 1388 | T02 | 01388 | 1. 00836<br>2. 00837 | 1. Host: Extended SBCS<br>2. Host: DBCS including 1880 user-defined characters |
| 9127 | T02 | 09127 | 1. 00836<br>2. 00837 | 1. Host: Extended SBCS<br>2. Host: DBCS including 1880 user-defined characters |

## Traditional Chinese

A list of the traditional Chinese ASCII and EBCDIC.

*Table 72. Traditional Chinese ASCII*

| CLINTCP | in | CCSID | CPGID | Comments |
|---|---|---|---|---|
| 938 | Base | 00938 | 1. 00904<br>2. 00927 | 1. PC data: SBCS<br>2. PC data: DBCS including 6204 user-defined characters |
| 948 | Base | 00948 | 1. 01043<br>2. 00927 | 1. PC data: Extended SBCS<br>2. PC data: DBCS including 6204 user-defined characters |
| 950<br>BIG5 | Base | 00950 | 1. 01114<br>2. 00947 | 1. PC data: SBCS (IBM BIG5)<br>2. PC data: DBCS including 13493 CNS, 566 IBM selected, 6204 user-defined characters |
| 964<br>EUCTW | Base | 00964 | 1. 00367<br>2. 00960<br>3. 00961 | 1. G0: ASCII<br>2. G1: CNS 11643 plane 1<br>3. G1: CNS 11643 plane 2 |
| 1370 | T02 | 01370 | 1. 01114<br>2. 00947 | 1. PC data: Extended SBCS; with euro<br>2. PC data: DBCS including 6204 user-defined characters; with euro |

*Table 73. Traditional Chinese EBCDIC*

| SRVERCP | in | CCSID | CPGID | Comments |
|---|---|---|---|---|
| 937 | Base | 00937 | 1. 00037<br>2. 00835 | 1. Host: Extended SBCS<br>2. Host: DBCS including 6204 user-defined characters |
| 1371 | T02 | 01371 | 1. 01159<br>2. 00835 | 1. Host: Extended SBCS; with euro<br>2. Host: DBCS including 6204 user-defined characters; with euro |

## Vietnamese

A list of the Vietnamese client and server CCSIDs.

*Table 74. 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 75. 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 IBM z Systems 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 76. 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 |

# Chapter 110. Bidirectional data support

CICS Transaction Gateway supports right-to-left bidirectional (bidi) data.

CICS Transaction Gateway can convert bidi data stored in CHAR containers from logical order to visual right-to-left order before sending this data to CICS. Bidi data returned from CICS is then also converted from visual right-to-left order to logical order. Bidi support is only available for CHAR containers.

The data in the CHAR container can be a mixture of text strings written in languages that are displayed left-to-right and right-to-left. If bidi data support is enabled and a CHAR container contains mixed language text strings, the conversion ensures that correct visual order for each language is applied. For details about how to configure right-to-left bidi text support, see Chapter 45, "Configuring bidirectional data support," on page 225.

# Part 17. Appendixes

# Glossary

This glossary defines the terms and abbreviations used in CICS Transaction Gateway and in the documentation.

**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*.

**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.

**Authorized Program Analysis Report (APAR)**
A request for correction of a defect in a current release of an IBM-supplied program.

**autoinstall**
A method of creating and installing resources dynamically as terminals log on, and deleting them at logoff.

**automatic restart manager (ARM)**
An IBM 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**

**Basic Mapping Support**
Basic mapping support is an interface between CICS and CICS application programs that move 3270 data streams to and from a terminal. The format of the input and output display data is defined by the BMS commands.

**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).

**BMS** see Basic Mapping Support

**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*.

**callback**
A way for one thread to notify another application thread that an event has happened.

**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.

**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**
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 IBM 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.

**CTIN** The CTIN transaction is invoked by the Client daemon to install a Client terminal definition on the CICS server.

## 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 Oracle 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 IBM 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 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 IBM 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. For more information, see Chapter 20, "File path terminology," on page 57.

**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.

**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.

**IPIC** See *IP interconnectivity*.

## J

**Java** An object-oriented programming language for portable interpretive code that supports interaction among remote objects.

**Java 2 Platform, Enterprise Edition (J2EE, Java EE)**
An environment for developing and deploying enterprise applications, defined by Oracle. 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 Oracle, 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 Oracle provided for the Java platform.

**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).

**JavaScript Object Notation (JSON)**
A lightweight data-interchange format that is based on the object-literal notation of JavaScript. JSON is programming-language neutral but uses conventions from languages that include C, C++, C#, Java, JavaScript, Perl, Python.

**JCA** See *JEE Connector Architecture*.

**JDK** See *Java development kit*.

**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).

**JNI** See *Java Native Interface*.

**JRE** See *Java Runtime Environment*.

**JSON** See *JavaScript Object Notation (JSON)*.

**JSON Schema**
A JavaScript Object Notation (JSON) document that describes the structure and constrains the contents of other JSON documents.

**JSON web service**
A web service that accepts and produces JSON payloads.

**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. 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 IBM 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 a user accesses the SNA network to communicate with another user and through which the 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 users, or between a 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 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*.

**OSGi** A specification that describes a modular system and a service platform for the Java programming language that implements a complete and dynamic component model.

**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.

**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)*.

**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.

**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 IBM 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*.

**<product_data_path>**
This term represents the directory used by the Windows CICS Transaction Gateway for common application data. For more information, see Chapter 20, “File path terminology,” on page 57.

**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 IBM 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, 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)**
An IBM z/OS facility that provides two-phase sync point support across participating resource managers.

**RESTful**
Pertaining to applications and services that conform to Representational State Transfer (REST) constraints.

**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.

**RPC** See *remote procedure call*.

**RRMS**
See *Recoverable resource management services*.

**RRS** See *Resource Recovery Services*.

**RU** See *Request unit* and *Response unit*.

**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.

**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.

**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.

**SIT** See *system initialization table*.

**SLU** See *secondary logical unit*.

**SMF** The IBM z/OS System Management Facility (SMF) collects and records system and job-related information that your IBM z/OS installation can use for reporting, billing, analysis, profiling, and maintaining system security. CICS TG for IBM z/OS writes statistical data to SMF.

**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.

**standard error**
In many workstation-based operating systems, the output stream to which error messages or diagnostic messages are sent.

**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.

**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 IBM z/OS) from the **ctgadmin** command (on UNIX, Linux, or Windows) or the IBM 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 IBM z/OS console.

**System Management Interface Tool (SMIT)**
An interface tool of the IBM 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 IBM 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 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**.

**WSBind file**
A Web service bind file is a resource that describes the specifics of the Web service.

**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).

# Related literature

Other documentation relating to CICS Transaction Gateway.

IBM Redbooks titles are available on a wide range of subjects relevant to CICS Transaction Gateway programming, installation, operation and troubleshooting. See the: IBM Redbooks site for more information.

Documentation for many IBM products is available online from the IBM Publications Center.

# 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 is compatible with the JAWS screen reader. 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 from a command prompt, specifying the -i console option.

Console mode displays text over multiple screens, and enables you to make choices during the installation process. The command prompt interface does not provide a cursor for navigating over the displayed text. When you use the JAWS screen reader you can repeat the displayed text with the command used for reading the current window (key combination Insert+B).

The first screen displayed by the installer is for language selection; the default language depends on the values in the system regional settings. To bypass the language selection screen, use the -l *lang* command option; where *lang* is one of the following:

- `de` German
- `en` English
- `es` Spanish
- `fr` French
- `it` Italian
- `ja` Japanese
- `ko` Korean
- `tr` Turkish
- `zh_CN` Chinese

For example, to install with the console interface in French:

```
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.

## 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'.
```

When using a screen reader on Windows, the Gateway daemon should be started and stopped with Windows services by starting and stopping the IBM CICS Transaction Gateway service. To determine if the CICS Transaction Gateway has started or stopped use the Windows Event Log viewer to check the messages in the Application log.

## 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, for more information, see *Keyboard mapping for cicsterm*.

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.

# Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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 grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing*
*IBM Corporation*
*North Castle Drive, MD-NC119*
*Armonk, NY 10504-1785*
*US*

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing*
*Legal and Intellectual Property Law*
*IBM Japan Ltd.*
*19-21, Nihonbashi-Hakozakicho, Chuo-ku*
*Tokyo 103-8510, Japan*

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 jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those

websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM Director of Licensing*
*IBM Corporation*
*North Castle Drive, MD-NC119*
*Armonk, NY 10504-1785*
*US*

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

The client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBMproducts. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the

names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.



## Programming interface information

## Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

## Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

### Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

### Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

### Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

### Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

## IBM Online Privacy Statement

## Safety and environmental notices

## Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

Intel is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

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

# Readers' Comments — We'd Like to Hear from You

**CICS Transaction Gateway for Multiplatforms**
**Version 9 Release 2**
**Multiplatform Administration**

**Publication No. SC34-7338-00**

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**
SC34-7338-00

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

SC34-7338-00

Cut or Fold
Along Line

SC34-7338-00