CICS Transaction Gateway
Version 9 Release 0

z/OS Administration

SC34-2832-02

CICS Transaction Gateway
Version 9 Release 0

z/OS Administration

SC34-2832-02

Note

Before using this information and the product it supports, read the information in “Notices” on page 387.

This edition applies to Version 9.0 of the CICS Transaction Gateway for z/OS program number 5655-Y20 and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Chapter 9. Deploying applications . . 169

Chapter 10. Scenarios. 177

About this information

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

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

What's new in CICS Transaction Gateway V9.0

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

Functional enhancements

- As an aid to migrating from EXCI to IPIC, an ECI timeout value can be set on IPIC server definitions in the CICS Transaction Gateway configuration file. This enables applications, that do not specify a timeout value, to be timed out when connected to CICS over IPIC.
- All ECI V2 and ESI V2 functions can now be used by 64-bit applications as well as by 32-bit applications. For more information, see *CICS TG ECI and ESI Version 2 for C Overview* in the online Information Center.
- ECI V2 applications now allow multiple threads to share a single Gateway connection when using synchronous calls.
- ECI V2 has been enhanced with the addition of asynchronous call support, for more information, see *Programming Guide*.
- Local mode JEE and Java™ applications can now set the maximum number of send sessions that are used with IPIC connections to CICS. The number of send sessions controls the number of simultaneous requests that are allowed over the connection.

High availability and scalability

- CICS Transaction Gateway can now operate as a 31-bit or 64-bit address space. With a 64-bit address space, the Gateway daemon can process workloads with larger channels and can use more connection manager and worker threads without exhausting address space resources. .
- When using the CICS request exit, the restrictions on the routing of XA transactions have been removed. Similar to SYNCONRETURN and extended LUW transactions, XA transactions can now be routed based on Transid, Program, and PayloadType values, for example.
- Client application IP addresses are now available for use in CICS request exits. The IP address information can be used in your CICS request exit routing decisions, for example.

Monitoring

- You can use Cross Component Trace (XCT) to track individual requests as they flow from WebSphere® Application Server to CICS and back, making it easier to identify the responsible component when failures occur. XCT can be enabled for applications using the ECI resource adapter in WebSphere Application Server V8.5 or later.
- Request monitoring exits in the Gateway daemon have been extended to include an exit that reports the actual target CICS server for a request. A new exit point, RequestDetails, is called after any DSS routing decisions have been made but before the request is sent to CICS. For more information, see the *Programming Guide*.
- Channel and container information is now available to Java request monitoring exits for ECI requests that have an associated channel. For more information, see the *Programming Guide*.

- The CTGSMFRD sample has been updated so that it can read and format combinations of SMF records that have been generated by any release of CICS Transaction Gateway from V7.1 or later. For more information, see the *Programming Guide*.
- Assembler DSECTs for the CICS TG SMF111 records have been provided, in library SCTGMAC, to simplify the use of SMF monitoring records by assembler programs.
- The Gateway daemon start time is now output in SMF records, you can use this to help identify a specific Gateway daemon instance from the SMF data.

Security

- The Gateway daemon now provides IPIC connections to CICS using SSL.
- Exclusion of weaker ciphers can now be enforced on SSL connections using the JSSE SP800-131A transition support. For more information, see SP800-131A Compliance.
- Password phrases can now be used for user ID authentication when using EXCI connections to CICS.
- CICS TG jars are now signed for compatibility with the January 2014 CPU (Oracle 7u51, IBM 7 SR6-FP1). For more information, see Signing Applets and Web Start Applications.

Globalization

- CICS Transaction Gateway has been updated to support right-to-left bidirectional (bidi) data in containers.

For other changes from previous releases see the Upgrading section.

Chapter 1. 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.

CICS Transaction Gateway also provides flexible deployment options for different architectures, for example:

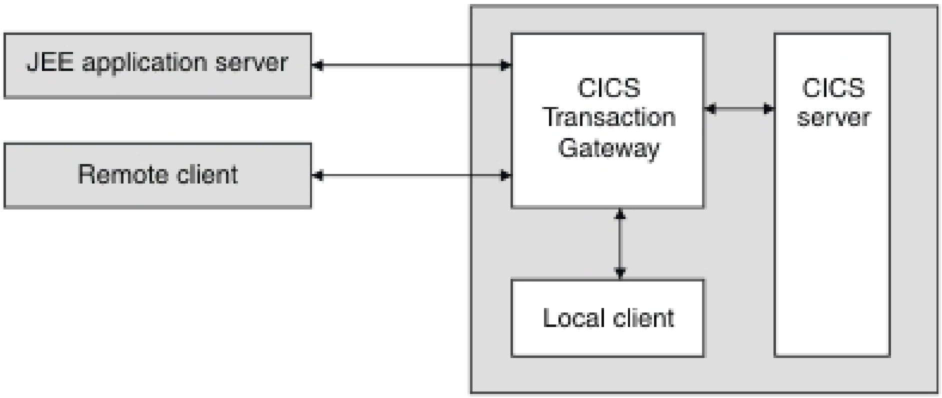


Figure 1. Access to CICS using CICS Transaction Gateway

On all operating platforms, CICS Transaction Gateway provides a gateway to CICS for remote clients and also complements IBM WebSphere Application Server on a range of different platforms. In addition, CICS Transaction Gateway for z/OS® is designed to exploit the qualities of service of the z/OS platform, including high availability and workload management.

CICS Transaction Gateway offers these features and benefits:

- A simple programming model with minimal change to CICS programs
- Access to COMMAREA, channel and 3270 applications
- A rich set of client 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 information center, see Glossary..

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 and C programming languages. JCA resource adapters, and a .NET 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. Applications developed in C++ and COBOL languages can call the C API.

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 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*.

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 concepts:

"Configuring CICS server connections" on page 110
The connections that CICS Transaction Gateway uses when forwarding client requests to CICS must be configured for the supported protocol.

Related information:

"Configuring EXCI" on page 119
The configuration of EXCI server connections is the same for both local and remote modes. A server definition is not required in the configuration file (ctg.ini) for remote mode. EXCI configuration is achieved through setting of environment variables.

"Which protocol can be used?" on page 14
This table shows what support is available for connecting to different version CICS servers over IPIC and EXCI.

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 either TCP/IP or SSL.

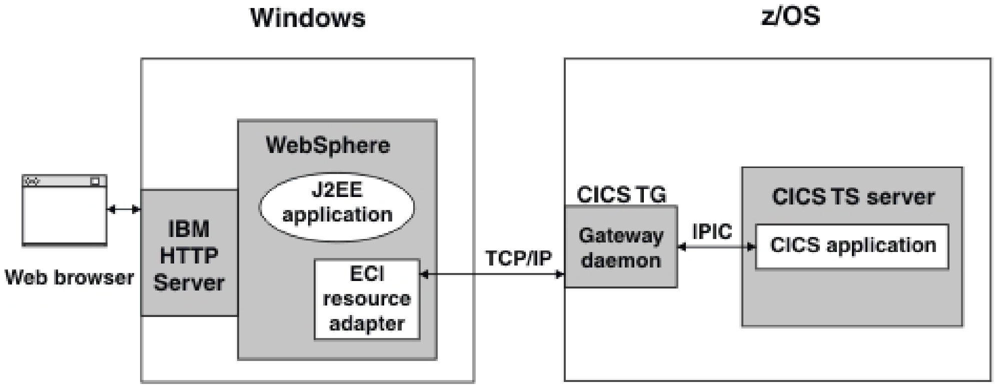


Figure 2. An example of CICS Transaction Gateway for z/OS in remote mode

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.

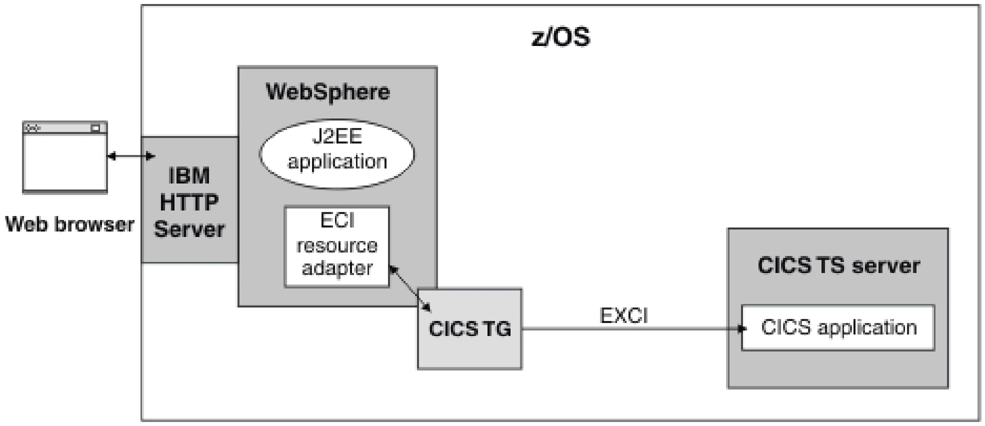


Figure 3. An example of CICS Transaction Gateway for z/OS in local mode

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 can be used for ECI requests to CICS COMMAREA or channel-based programs. IPIC supports two-phase commit transactions. SSL can be configured on IPIC connections in a local mode (two-tier) configuration. IPIC also supports the offload of work to a zSeries Application Assist Processor (zAAP).
EXCI	This protocol can be used for ECI requests to CICS COMMAREA programs. EXCI also supports two-phase commit transactions.

High availability

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

A high availability scenario can be implemented in remote mode using one or more of these ways:

- TCP/IP load balancing where Gateway daemons use TCP/IP port sharing or the IBM Sysplex Distributor
- Health reporting

- Dynamic server selection

CICS Transaction Gateway can exploit the high availability, load balancing, and workload management characteristics of CICS Transaction Server for z/OS. CICS Transaction Gateway also provides integration with the Internet Protocol (IP) workload management functions of z/OS, including IBM z/OS Workload Manager (WLM) and IBM Sysplex Distributor.

TCP/IP load balancing

TCP/IP load balancing can be achieved using TCP/IP port sharing or the Sysplex Distributor. If TCP/IP load balancing is used, connections are shared between several Gateway daemons listening on a single TCP/IP port and creating a highly available Gateway group. This provides a single end point for client requests, regardless of the type of transaction being handled.

Health reporting is an additional mechanism that can be used to influence the TCP/IP load balancing algorithm so that unavailable CICS servers are not used.

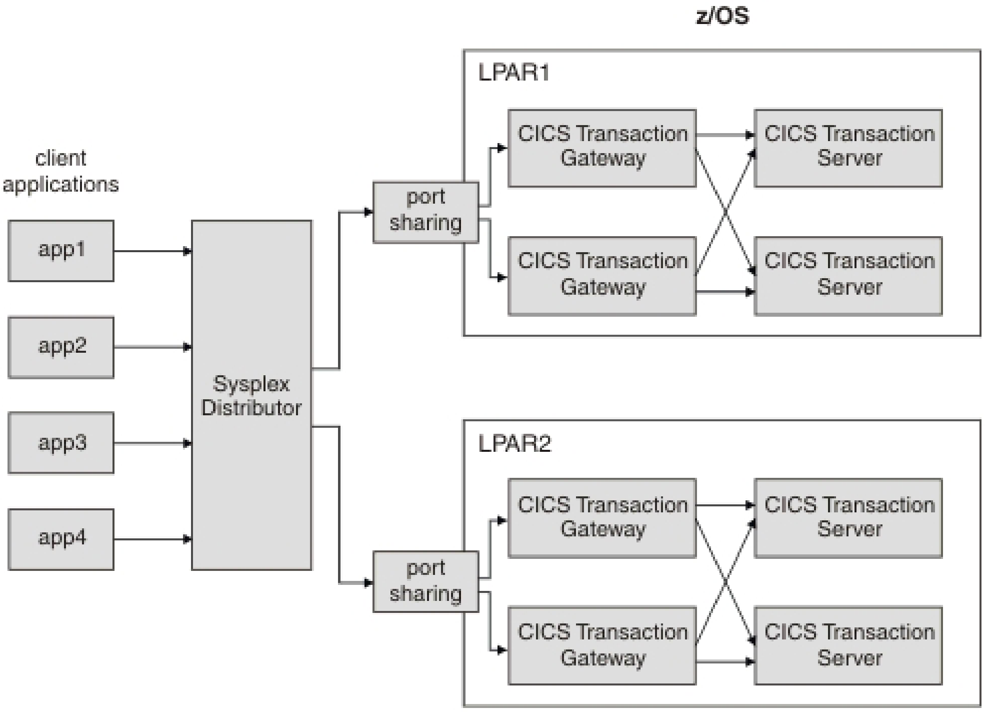


Figure 4. High availability

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. Two-phase commit XA transactions are eligible for dynamic server selection.

Security

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

Network security

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

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

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

User authentication

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

Link security

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

Bind security

Bind security prevents an unauthorized remote system from connecting to CICS. Bind security can be implemented for the IPIC and EXCI protocols.

Statistics and monitoring

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

Statistics

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

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

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

Monitoring

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

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

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

Tooling and product integration

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

- IBM Rational® Application Developer provides a J2C toolkit. The toolkit enables you to generate code for use with the CICS Transaction Gateway ECI resource 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.
- IBM CICS Performance Analyzer provides statistics alert reporting. You can define conditions, in terms of CICS Transaction Gateway statistics values, that interest you. You can then use those conditions to report on statistics stored in SMF records.
- IBM Tivoli System Automation for z/OS enables alerts to be generated based on CICS Transaction Gateway messages written to the z/OS console. For more information see “System automation messages” on page 248.
- IBM Tivoli Composite Application Manager can use CICS Transaction Gateway request monitoring exits to obtain performance information, and to analyze composite transactions occurring between a JEE application server and CICS.

Chapter 2. Planning

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

For information about upgrading from an earlier version of CICS Transaction Gateway, see Chapter 4, “Upgrading,” on page 25.

Hardware requirements

CICS Transaction Gateway requires an IBM System z® mainframe.

Supported software

CICS Transaction Gateway (CICS TG) 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, utilities installed with the product, and applications that use the Java API in local mode.

Operating systems

The operating systems that CICS Transaction Gateway supports.

CICS Transaction Gateway for z/OS can be used with z/OS V1.12 and later.

Java runtime environments for core components

The Java runtime environments that can be used with the Gateway daemon and local mode applications.

- IBM 31-bit SDK for z/OS, Java Technology Edition V7.0 SR1, and later service refreshes
- IBM 64-bit SDK for z/OS, Java Technology Edition V7.0 SR1, and later service refreshes

CICS servers

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

- CICS Transaction Server for z/OS V3.1 and later
- TXSeries for Multiplatforms V7.1

Note: CICS Transaction Server for z/OS V4.1 requires APARs PK83741 and PK95579 for identity propagation support.

JEE application servers

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

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

WebSphere Application Server for Multiplatforms⌂ is supported in remote mode only⌂.⌂

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

Supported in local and remote modes.

WebSphere Application Server for Multiplatforms⌂ V7.0

Supported in remote mode only⌂, using the CICS Transaction Gateway V8.0 or earlier JCA 1.5 resource adapters connecting to CICS Transaction Gateway V9.0⌂.

WebSphere Application Server for z/OS V7.0

Supported in remote mode only⌂, using the CICS Transaction Gateway V8.0 or earlier JCA 1.5 resource adapters connecting to CICS Transaction Gateway V9.0⌂.

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.

JEE 5 application servers

Any JEE 5 or above certified application server that successfully runs the JCA resource adapter IVT. ⌂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. ⌂ Supported in remote mode only, using the CICS Transaction Gateway V8.0 or earlier JCA 1.5 resource adapters connecting to CICS Transaction Gateway V9.0⌂.

Note:

1. The CICS Transaction Gateway V8.0 and earlier 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 165.

Java application runtime environments

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

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.⌂

Java SE 5

All IBM provided Java SE environments are supported. ⌂ Any Java 5 compatible SDK carrying the Java Compatible logo is supported.⌂

Note:

1. 32-bit (31-bit on z/OS) 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).

Redistributable component runtime environments

A number of runtime environments can be used with CICS Transaction Gateway redistributable components (the ECI V2, ESI V2, Statistics and .NET APIs).

AIX®

XL C/C++ V10.1, V11.1, and V12.1 compatible runtimes are supported on AIX V6.1 and later.

HP-UX

aC++ compatible runtime⌂s and ANSI C compatible runtimes are supported on:

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

Linux on Intel

XL C/C++ V10.1, V11.1, and V12.1 compatible runtime⌂s and gcc 4.1 compatible runtimes are supported on:

- SuSE Linux Enterprise Server 10
- SuSE Linux Enterprise Server 11
- SuSE Linux Enterprise Desktop 10
- SuSE Linux Enterprise Desktop 11
- Red Hat Enterprise Linux V5
- Red Hat Enterprise Linux V6
- Red Hat Enterprise Linux compatible environments (see note 2 on page 12)
- Red Hat KVM with Red Hat Enterprise Linux (RHEL) operating system

- SUSE KVM with SUSE Linux Enterprise Server (SLES) operating system

Linux on POWER®

XL C/C++ V10.1, V11.1, and V12.1 compatible runtimes and gcc 4.1 compatible runtimes are supported on:

- SuSE Linux Enterprise Server 10
- SuSE Linux Enterprise Server 11
- Red Hat Enterprise Linux V5
- Red Hat Enterprise Linux V6
- Red Hat KVM with Red Hat Enterprise Linux (RHEL) operating system
- SUSE KVM with SUSE Linux Enterprise Server (SLES) operating system

Linux on System z

XL C/C++ V10.1, V11.1, and V12.1 compatible runtime⌂s and gcc 4.1 compatible runtimes are supported on:

- SuSE Linux Enterprise Server 10
- SuSE Linux Enterprise Server 11
- Red Hat Enterprise Linux V5
- Red Hat Enterprise Linux V6
- Red Hat KVM with Red Hat Enterprise Linux (RHEL) operating system
- SUSE KVM with SUSE Linux Enterprise Server (SLES) operating system

Solaris

Oracle Solaris Studio 12.3 compatible runtime⌂s are supported on:

- Solaris 10

Windows

.NET Framework 4.0,⌂ .NET Framework 3.5⌂, and Visual Studio 2010 Runtime Libraries are supported in 32-bit and 64-bit environments on:

- Windows 8
- Windows Server 2012
- Windows 7
- Windows Server 2008 R2⌂
- Windows Server 2008
- Windows Vista⌂

Note:

1. CICS Transaction Gateway does not support security-enhanced Linux.
2. Red Hat Enterprise Linux compatible environments must be binary and source compatible with a Red Hat Enterprise Linux version listed on this page as supported for Linux on Intel.
3. Runtime environments later than those listed are supported, if the compiler vendor provides upward compatibility.⌂

Development environments

The CICS Transaction Gateway APIs support a number of development environments.

Windows

- Microsoft Visual Studio 2012⌂
- Microsoft Visual Studio 2010⌂
- .NET Framework 4.5
- .NET Framework 4.0
- .NET Framework 3.5

AIX

- XL C/C++ for AIX V12.1
- XL C/C++ for AIX V11.1⌂
- XL C/C++ for AIX V10.1⌂

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, V12.1
- XL C/C++ for Linux, V11.1⌂
- XL C/C++ for Linux, V10.1⌂
- gcc 4.1 compatible runtime⌂ for Linux

Java

All IBM provided Java SE environments are supported. ⌂Any Java 7, Java 6 or Java 5 compatible SDK carrying the Java Compatible logo is supported. ⌂

- Java SE 7⌂
- Java SE 6⌂
- Java SE 5⌂

Note:

1. Development using the .NET API is supported only on Windows.
2. C++ 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

CICS Transaction Gateway includes a plug-in for CICS Explorer.

- CICS Transaction Gateway plug-in for CICS Explorer V1.1 is supported in CICS Explorer V1.1 and V1.1.1

- CICS Transaction Gateway plug-in for CICS Explorer V2.0 is supported in CICS Explorer V5.1.
- CICS Transaction Gateway plug-in for CICS Explorer V2.0.1 is supported in CICS Explorer V5.1.1.

Applications compiled for earlier versions

Applications that are compiled for use with earlier versions of CICS Transaction Gateway, and which are dynamically linked to the API, are supported, and do not have to be recompiled.

Which protocol can be used?

This table shows what support is available for connecting to different version CICS servers over IPIC and EXCI.

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

Table 1. IPIC and EXCI communication with CICS servers

CICS server	IPIC	EXCI
CICS Transaction Server for z/OS V5.1	XA, ECI COMMAREA and channels, ESI, password phrases	XA, ECI COMMAREA, password phrases
CICS Transaction Server for z/OS V4.2	XA, ECI COMMAREA and channels, ESI, password phrases	XA, ECI COMMAREA, password phrases
CICS Transaction Server for z/OS V4.1	XA, ECI COMMAREA and channels, ESI	XA, ECI COMMAREA, password phrases
CICS Transaction Server for z/OS V3.2	XA, ECI COMMAREA and channels	XA, ECI COMMAREA, password phrases
CICS Transaction Server for z/OS V3.1	Not supported	XA, ECI COMMAREA, password phrases
TXSeries for Multiplatforms V7.1	ECI COMMAREA and channels	Not supported

Which API can be used?

This table shows which APIs are supported over the IPIC and EXCI protocols in local and remote mode.

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

API	IPIC local mode	IPIC remote mode	EXCI local mode	EXCI remote mode
Java ECI	✓ (see Note)	✓ (see Note)	✓	✓
Java ESI	✓	✓	x	x
JEE non-XA	✓ (see Note)	✓ (see Note)	✓	✓
JEE XA	✓ (see Note)	✓ (see Note)	✓	✓
C ECI V2	x	✓ (see Note)	x	✓
C ESI V2	x	✓	x	x

API	IPIC local mode	IPIC remote mode	EXCI local mode	EXCI remote mode
.NET ECI	x	✓ (see Note)	x	✓
.NET ESI	x	✓	x	x

Note: Includes channel application support. All ECI APIs and protocols support COMMAREA based applications.

Sysplex restrictions

Restrictions apply when you are using CICS Transaction Gateway in a sysplex environment.

The CICS Transaction Gateway instance must run on the same LPAR as the CICS server to which it is sending requests if you are using EXCI connections with one of the following:
- The ECI resource adapter for global transactions
- A Client application for extended LUW requests

You can start a Gateway daemon in an HA group in any LPAR in the sysplex but you must ensure that the same RRS logging group is used in each LPAR.

Using multiple releases of CICS TG

You can run multiple versions of the Gateway in the same LPAR. To enable this when using XA support, the CTGRRMS services must be running, and must be at the level of the latest release of CICS Transaction Gateway installed on the LPAR.

You must use ctgasi or a Gateway from the latest release to start the CTGRRMS services before you start any of the Gateways from the earlier releases. The CTGRRMS service task stays active for the duration of the z/OS image. Therefore, starting or refreshing the services must be done when the new release of the Gateway is installed, or following a re-IPL of the LPAR.

Check that the LLA list contains the latest level of the SCTGLINK data set by using this command:

```
DISPLAY LLA
```

and see “Refreshing CTGRRMS services” on page 129 if you need to refresh CTGRRMS services.

If you need to re-IPL an LPAR, perform the procedure described in “LPAR IPLs in an XA environment” on page 131.

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 application compatibility

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

Client applications built using an earlier version of the CICS Transaction Gateway .NET API can connect to a V9.0 Gateway daemon by using the CICS Transaction Gateway .NET API assembly that they were compiled against.

To use the new CICS Transaction Gateway V9.0 .NET API assembly, client applications that were built using the CICS Transaction Gateway V8.0 .NET API and target Microsoft .NET Framework V4.0 must be recompiled. Client applications that target Microsoft .NET Framework V3.5 or were built using the CICS Transaction Gateway V8.1 .NET API can use the new CICS Transaction Gateway V9.0 .NET API by using assembly redirection; these applications do not require recompiling.

Assembly redirection can be configured at the application or machine level by adding the following code to the application configuration file, web 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="9.0.0.0" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

Replace “x.x” with the version of the CICS Transaction Gateway .NET API that the application was compiled against.

If the CICS Transaction Gateway .NET API assembly is installed in the Global Assembly Cache (GAC), assembly redirection can be configured by installing the appropriate publisher policy file into the GAC. Policy files are provided in *install_path*`/Windows/lib/policy` in the ctgredist package or in *install_path*`/lib/policy` on a Windows machine with CICS Transaction Gateway installed. For more information on publisher policy and assembly redirection, see the Microsoft .NET Framework documentation.

For information on upgrading .NET applications from CICS Transaction Gateway V8.0, see “CICS Transaction Gateway .NET applications” on page 28.

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. For more information see Chapter 4, “Upgrading,” on page 25.

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.

National Language Support

CICS Transaction Gateway supports the same languages as CICS Transaction Server for z/OS: English, Japanese and Simplified Chinese.

For information about code pages see “Supported code pages” on page 22.

Tools

Tools that support CICS Transaction Gateway.

- IBM CICS Performance Analyzer
- IBM Tivoli System Automation for z/OS
- IBM Tivoli OMEGAMON XE for CICS on z/OS
- IBM Tivoli Composite Application Manager for Transactions
- IBM Rational Developer for System z
- IBM Rational Application Developer for WebSphere Software
- IBM Rational Business Developer

- IBM Rational Software Architect for WebSphere Software
- IBM Integration Designer

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

Chapter 3. Installing

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

CICS Transaction Gateway runs in a nonswappable address space to prevent ECI calls from failing if they are flowed when CICS Transaction Gateway is swapped out. Ensure that the user who is running the CICS TG process has READ access to the BPX.STOR.SWAP FACILITY class. For information about the CTG_SWAPPABLE environment variable that controls this setting see "Environment variables: local and remote mode" on page 100.

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.

The <install_path> is `/usr/lpp/cicstg/ctgversion` where *ctgversion* is the product version, and can be modified at installation time. A lowercase convention is normally used for file paths.

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`.

High level qualifier

The term *hlq* indicates the high-level qualifier used for MVS data sets, for example `hlq.SCTGLINK`. The high level qualifier can be set during the SMP/E installation, for example, if <install_path> is /usr/lpp/cicstg/ctg900/bin, you could set the *hlq* to CICSTG.CTG900.

Actions after installation

When you have installed the CICS Transaction Gateway, configure and test your installation.

To configure and test your installation follow these steps:

1. Configure your CICS Transaction Server for z/OS; see "Configuring CICS server connections" on page 110.
2. Configure your CICS Transaction Gateway by creating these files:
   - ctg.ini (required only if running in remote mode)
   - ctg.env or ctgenvvar

   See "Configuration parameter reference" on page 156.
3. Test your configuration; see Testing your configuration.

You must also configure READ access to the BPX.SMF facility as part of installation and upgrading. The RACF and UNIX System Services (USS) documentation gives further details about this permission.

An attempt is always made to write end-of-day and shutdown statistics to SMF, regardless of whether the **`statsrecording`** parameter is on or off. A message is logged if this attempt fails.

To write to SMF, the user ID under which the Gateway daemon runs, must be permitted READ access to the BPX.SMF facility. An example of the syntax is shown here:

```
PERMIT BPX.SMF CLASS(FACILITY) ACCESS(READ) ID(USERID)
```

Changing the code page

You can convert your CICS Transaction Gateway installation to use a different EBCDIC code page if required, for example to change the language.

If the product was installed as IBM-1047 or another code page (code set), you can convert the installation to use a different code page by running the following command in the current code page of the installation:

```
<install_path>/bin/ctg2local
```

You need write access to the directory in which the CICS Transaction Gateway is installed to run this command.

Converting the installation to another code page does not affect messages used by the CTGBATCH program. See "CTGBATCH considerations" on page 250 for details.

To display the supported language and code set combinations, issue the command:

```
<install_path>/bin/ctgmsgs -?
```

Supported code pages

The code ages supported for US English, Simplified Chinese and Japanese.

Language	Supported code pages
US English	IBM-1047
Simplified Chinese	IBM-935 IBM-1388 IBM-9127
Japanese	IBM-930 IBM-939 IBM-1930 IBM-1939

Redistributable components

A redistributable package is available that contains the components required for developing and running remote C and .NET applications.

The ctgredist package is located in the <install_path>/deployable directory and provides components for developing remote C and .NET applications that access

CICS servers using the CICS Transaction Gateway. Runtime libraries are provided for application deployment on systems remote to the CICS Transaction Gateway.

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

Chapter 4. Upgrading

Use this information to plan and upgrade your environment and applications to work in the latest release of CICS Transaction Gateway.

Before you begin your upgrade, check if there are any compatibility issues for your applications or system resources. For more information, see Compatibility.

Upgrading from Version 8 Release 1

This information details changes that you need to consider when upgrading to the current release from V8.1.

Java Version 7

Ensure the PATH environment variable in the STDENV file used during CICS TG initialization contains the location of the IBM Java 7 runtime environment.

Removal of Gateway daemon resource parameters

The **`uowvalidation`** parameter is longer supported. This means that LUW tokens are validated so that they can be used only from the client application connection from which the LUW was started.

If the **`uowvalidation`** parameter is specified in the configuration file, the Gateway daemon fails to start. You must remove the **`uowvalidation`** parameter from the configuration file.

If any of your applications rely on the **`uowvalidation`** parameter being turned off, the application receives the ECI_ERR_LUW_TOKEN error when trying to use the LUW token on a connection, to the Gateway daemon, that did not start the LUW.

Removal of ciphersuites=128bitonly parameter

Use of the **`ciphersuites=128bitonly`** parameter is deprecated.

Removal of -noshareclasses option

The undocumented `-noshareclasses` option has been removed. If it is specified, the Gateway daemon fails to start and the following message is written to STDERR:

```
CTG6582E The command line option -noshareclasses is unknown or requires a
value
```

If there is a requirement to disable Java class caching, the documented Java argument `-Xshareclasses:none` can be used.

SSL keyring settings moved

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: **esmkeyring**, **hwcrypt**, **keyring**, **keyringpw**, and **keyringpwscrambled**. The **esmkeyring** and **hwcrypt** parameters now take a parameter value when defined in the PRODUCT section.

TLS cipher suites

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

IPIC server idle timeout default setting

The default setting of the server idle timeout period for IPIC server connections (see "Server idle timeout" on page 117) 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.

Message_Qualifier API removal

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 V9.0 when run in remote mode using a CICS TG V8.1 or earlier `ctgclient.jar` file

Upgrading from Version 8 Release 0

This information details changes that you need to consider when upgrading to the current release from V8.0.

Java Version 7

Ensure the PATH environment variable in the STDENV file used during CICS TG initialization contains the location of the IBM Java 7 runtime environment.

Removal of Gateway daemon resource parameters

The **uowvalidation** parameter is longer supported. This means that LUWs tokens are validated so that they can be used only from the client application connection from which the LUW was started.

If the **uowvalidation** parameter is specified in the configuration file, the Gateway daemon fails to start. You must remove the **uowvalidation** parameter from the configuration file

If any of your applications rely on the **uowvalidation** parameter being turned off, the application receives the ECI_ERR_LUW_TOKEN error when trying to use the LUW token on a connection, to the Gateway daemon, that did not start the LUW.

Removal of ciphersuites=128bitonly parameter

Use of the **`ciphersuites=128bitonly`** parameter is deprecated.

SSL keyring settings moved

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: **`esmkeyring`**, **`hwcrypt`**, **`keyring`**, **`keyringpw`**, and **`keyringpwscrambled`**. The **`esmkeyring`** and **`hwcrypt`** parameters now take a parameter value when defined in the PRODUCT section.

TLS cipher suites

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

Removal of -noshareclasses option

The undocumented `-noshareclasses` option has been removed. If it is specified, the Gateway daemon fails to start and the following message is written to STDERR:

```
CTG6582E The command line option -noshareclasses is unknown or requires a
value
```

If there is a requirement to disable Java class caching, the documented Java argument `-Xshareclasses:none` can be used.

Using the JEE interfaces in nonmanaged mode

The JAR file cicsj2ee.jar file is renamed to cicsjee.jar.

Supported characters in server names

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 in Configuring CICS server connections.

IPIC server idle timeout default setting

The default setting of the server idle timeout period for IPIC server connections 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.

Message_Qualifier API removal

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.

CICS Transaction Gateway .NET applications

The CICS Transaction Gateway .NET API 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 ctgredist package in `Windows\lib`. The upgraded CICS Transaction Gateway .NET API 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 .NET API 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 .NET API 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 that target 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 `policy.8.0.IBM.CTG.Client.dll`, into the GAC. The assembly and the publisher policy assembly are both located in `<install_path>/Windows/lib/policy` in the `ctgredist` package or in on a Windows machine with CICS Transaction Gateway installed. 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="8.0.0.0-8.0.0.9" newVersion="9.0.0.0" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

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

Logical CICS server definitions

LOGICALSERVER section definitions are deprecated and superseded by policy-based dynamic server selection definitions.

A LOGICALSERVER section definition in the configuration file (ctg.ini), provides a mapping of a logical CICS server name to an actual CICS server name where work is to be sent. The logical CICS server name is an alias that can be passed to CICS Transaction Gateway on an ECI or ESI request. The function provided by LOGICALSERVER section definitions is incorporated into policy-based dynamic server selection and this new mechanism provides enhanced capability and flexibility for dynamic server selection.

Replace LOGICALSERVER section definitions with policy-based DSS definitions. For example the definitions:

```
SECTION LOGICALSERVER = PAYMONTH
Server = PAYROLLA
ENDSECTION

SECTION LOGICALSERVER = PAYWEEK
Server = PAYROLLB
ENDSECTION
```

can be replaced by the definitions:

```
SECTION GATEWAY
....
DSSPolicy = POLICY1
ENDSECTION

SECTION DSSPOLICY = POLICY1
SUBSECTION MAPPINGS
PAYMONTH=GROUP1
PAYWEEK=GROUP2
ENDSUBSECTION

SECTION DSSGROUP = GROUP1
Servers = PAYROLLA
Algorithm = Failover
ENDSECTION

SECTION DSSGROUP = GROUP2
Servers = PAYROLLB
Algorithm = Failover
ENDSECTION
```

When using LOGICALSERVER section definitions:

- If a request does not specify the name of a CICS server, CICS Transaction Gateway uses the default CICS server, if one has been defined. The same applies when using policy-based DSS, if a request does not specify the name of a CICS server and no server group mapping is defined.
- If a request does not specify a name and a default CICS server has not been defined, CICS Transaction Gateway assumes that the EXCI protocol is being used, and delegates the decision about which CICS server to use, to EXCI. The same applies when using policy-based DSS, if a request does not specify a name that matches any server group mapping and a default CICS server has not been defined.

For information about the LOGICALSERVER section of the configuration file see "LOGICALSERVER section of the configuration file" on page 162.

Configuring a logical CICS server

To configure a logical CICS server definition add a LOGICALSERVER section to the configuration file.

LOGICALSERVER section definitions are deprecated and are superseded by "Policy-based dynamic server selection" on page 77 definitions. For information about configuring a logical CICS server definition see "LOGICALSERVER section of the configuration file" on page 162.

Server name:

The LOGICALSERVER section parameter specifies a logical CICS server name that is local to CICS Transaction Gateway.

LOGICALSERVER section definitions are deprecated and are superseded by "Policy-based dynamic server selection" on page 77 definitions.

Set the value in the range of 1 to 8 characters. Supported characters are in the ranges A-Z and 0-9, and '@', '#', '$'."

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 161 and the .

Default value

There is no default value for this parameter.

CICS server name:

The name of the actual CICS server to which the logical server definition is to resolve.

This is either the name of an IPIC server definition or a CICS APPLID to be used for an EXCI connection. Specify an alphanumeric identifier, up to eight characters in length.

Upgrading from Version 7 Release 2

This information details changes that you need to consider when upgrading to the current release from V7.2.

Configuration files

CICS Transaction Gateway configurations that worked with previous releases might not work after 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.

The Configuration Tool is removed from this version of CICS Transaction Gateway for z/OS. To assist in configuring CICS Transaction Gateway additional scenario configuration files are provided. For more information, see Scenarios.

Java Version 7

Ensure the PATH environment variable in the STDENV file used during CICS TG initialization contains the location of the IBM Java 7 runtime environment.

Removal of Gateway daemon resource parameters

The **`uowvalidation`** parameter is longer supported. This means that LUW tokens are validated so that they can be used only from the client application connection from which the LUW was started.

If the **`uowvalidation`** parameter is specified in the configuration file, the Gateway daemon fails to start. You must remove the **`uowvalidation`** parameter from the configuration file.

If any of your applications rely on the **`uowvalidation`** parameter being turned off, the application receives the ECI_ERR_LUW_TOKEN error when trying to use the LUW token on a connection, to the Gateway daemon, that did not start the LUW.

Upgrading XA configurations

When you upgrade from Version 7 Release 2 and were previously using XA support over IPIC connections without using CTGRRMS you must enable CTGRRMS.

Determine whether you are using CTGRRMS by checking the **`xasupport`** parameter in the Gateway daemon configuration file.

- If **xasupport=off**, or if the configuration file does not contain the **`xasupport`** parameter, no upgrade steps are needed and the CICS Transaction Gateway will work as it did before the upgrade.
- If **xasupport=on**, you are using XA and you must follow the upgrade steps. For more information see “Configuring XA support” on page 126.

When you enable XA support, check the Gateway daemon log for message CTG6737I which indicates that XA support is enabled.

User exits

CICS request exit
All CICS request exits must be updated to support the eventFired() method. For more information, see the *CICS Transaction Gateway for z/OS Programming Guide*.

Request monitoring exit
The CICSCorrelator in the RequestData map must be used to report the CICS Network UOWID for EXCI synconreturn requests.

Using the JEE interfaces in nonmanaged mode

The JAR file cicsj2ee.jar file is renamed to cicsjee.jar.

Supported characters in server names

Server names must now use characters from the supported character list (see "Server name" on page 30) 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.

IPIC server idle timeout default setting

The default setting of the server idle timeout period for IPIC server connections 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.

Message_Qualifier API removal

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.

Removal of ciphersuites=128bitonly parameter

Use of the **`ciphersuites=128bitonly`** parameter is deprecated.

SSL keyring settings moved

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: **`esmkeyring`**, **`hwcrypt`**, **`keyring`**, **`keyringpw`**, and **`keyringpwscrambled`**. The **`esmkeyring`** and **`hwcrypt`** parameters now take a parameter value when defined in the PRODUCT section.

TLS cipher suites

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

Upgrading from Version 7 Release 1

This information details changes that you need to consider when upgrading to the current release from V7.1.

Configuration files

CICS Transaction Gateway configurations that worked with previous releases might not work after 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.

The Configuration Tool is removed from this version of CICS Transaction Gateway for z/OS. To assist in configuring CICS Transaction Gateway additional scenario configuration files are provided. For more information, see Scenarios.

Java Version 7

Ensure the PATH environment variable in the STDENV file used during CICS TG initialization contains the location of the IBM Java 7 runtime environment.

Removal of Gateway daemon resource parameters

The **`uowvalidation`** parameter is longer supported. This means that LUWs tokens are validated so that they can be used only from the client application connection from which the LUW was started.

If the **`uowvalidation`** parameter is specified in the configuration file, the Gateway daemon fails to start. You must remove the **`uowvalidation`** parameter from the configuration file.

If any of your applications rely on the **`uowvalidation`** parameter being turned off, the application receives the ECI_ERR_LUW_TOKEN error when trying to use the LUW token on a connection, to the Gateway daemon, that did not start the LUW.

Upgrading a statistics API port definition

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.

Upgrading XA configurations

When you upgrade from Version 7 Release 1 and were previously using XA support over IPIC connections without using CTGRRMS you must enable CTGRRMS.

Determine whether you are using CTGRRMS by checking the **`xasupport`** parameter in the Gateway daemon configuration file.

- If **xasupport=off**, or if the configuration file does not contain the **`xasupport`** parameter, no upgrade steps are needed and the CICS Transaction Gateway will work as it did before the upgrade.

- If **xasupport=on**, you are using XA and you must follow the upgrade steps. For more information see "Configuring XA support" on page 126.

When you enable XA support, check the Gateway daemon log for message CTG6737I which indicates that XA support is enabled.

When you upgrade from Version 7 Release 1 or earlier, if you have a Gateway daemon configuration using XA support, you must change the configuration settings. For more information see "Upgrading XA configurations" on page 35.

Using the JEE interfaces in nonmanaged mode

The JAR file cicsj2ee.jar file is renamed to cicsjee.jar.

Supported characters in server names

Server names must now use characters from the supported character list (see "Server name" on page 30) 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.

IPIC server idle timeout default setting

The default setting of the server idle timeout period for IPIC server connections 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.

Message_Qualifier API removal

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.

Removal of ciphersuites=128bitonly parameter

Use of the **`ciphersuites=128bitonly`** parameter is deprecated.

SSL keyring settings moved

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: **`esmkeyring`**, **`hwcrypt`**, **`keyring`**, **`keyringpw`**, and **`keyringpwscrambled`**. The **`esmkeyring`** and **`hwcrypt`** parameters now take a parameter value when defined in the PRODUCT section.

TLS cipher suites

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

Upgrading XA configurations

When you upgrade from Version 7 Release 1 or earlier, if you have a Gateway daemon configuration using XA support, you must change the configuration settings.

Analyze your existing configuration to determine the upgrade steps required

1. Determine whether or not you are using XA support by checking the **`xasupport`** parameter in the Gateway daemon configuration file.
   - If **`xasupport=off`**, or if the configuration file does not contain the **`xasupport`** parameter, no upgrade steps are needed and the CICS Transaction Gateway will work as it did before the upgrade.
   - If **`xasupport=on`**, you are using XA and you must follow the upgrade steps.

   You can also check the Gateway daemon log for message CTG6737I which indicates that XA support is enabled.
2. Analyze the configuration to determine if the Gateway daemon is part of a Gateway group by looking for the **`CTG_RRMNAME`** and **`CTG_MASTER_RRMNAME`** environment variables in the STDENV DD card of the JCL for the CTGBATCH job step. Older configurations might have set the same environment variables in a ctgenvvar script that is located on the HFS. You can also check the Gateway daemon log for message CTG9214I which indicates that the Gateway daemon is part of a Gateway group.
   - The Gateway is not configured to be part of a Gateway group if **`CTG_RRMNAME`** is specified and **`CTG_MASTER_RRMNAME`** is not specified.
   - The Gateway is configured to part of a Gateway group if both **`CTG_RRMNAME`** and **`CTG_MASTER_RRMNAME`** are specified. All Gateways that share the same **`CTG_MASTER_RRMNAME`** are in the same Gateway group.

Related reference:

"Using multiple releases of CICS TG" on page 15
You can run multiple versions of the Gateway in the same LPAR. To enable this when using XA support, the CTGRRMS services must be running, and must be at the level of the latest release of CICS Transaction Gateway installed on the LPAR.

Upgrading a Gateway with XA support

To upgrade a Gateway with XA support and not a member of a Gateway group.

About this task

1. Refresh the CTGRRMS services, follow the steps in "Refreshing CTGRRMS services" on page 129.
2. Remove the **`CTG_RRMNAME`** from the STDENV file.
3. Specify an **`APPLID`** and **`APPLIDQUALIFIER`**. The fully qualified APPLID must be unique within the sysplex.

   See Configuring identification using APPLID for a general explanation of **`APPLID`** and **`APPLIDQUALIFIER`**. See EXCI server connections for details about what **`APPLID`** and **`APPLIDQUALIFIER`** must be set to when XA support is enabled.
4. The Gateway registers with the following resource manager name: `CICSTG.<APPLIDQUALIFIER>.<APPLID>`.

5. You must permit access for the Gateway daemon USERID to one of the following RACF® facilities when XA support is enabled:
   - ALTER access to the MVSADMIN.RRS.COMMANDS.** facility.
   - ALTER access to the MVSADMIN.RRS.COMMANDS.gname.sysname facility.

   gname
   is the logging group name and corresponds to the logging group in the RRS administrative panels in ISPF. Set **gname** to the value for the sysplex where the Gateway is running.

   sysname
   is the system name. Set **sysname** to the value for the LPAR where the Gateway is running

   If you give ALTER access to MVSADMIN.RRS.COMMANDS.** the Gateway daemon is permitted to perform recovery operations for transactions associated with any system name or logging group. This option requires less administration but does not provide granularity of control.

   If you give ALTER access to MVSADMIN.RRS.COMMANDS.gname.sysname the Gateway daemon is permitted to perform recovery operations for transactions associated with the specified system name or logging group. This option allows for greater granularity of control but requires a greater amount of administration.

Related reference:

"Using multiple releases of CICS TG" on page 15
You can run multiple versions of the Gateway in the same LPAR. To enable this when using XA support, the CTGRRMS services must be running, and must be at the level of the latest release of CICS Transaction Gateway installed on the LPAR.

Upgrading a Gateway group with XA support

1. Refresh the CTGRRMS services, follow the steps in "Refreshing CTGRRMS services" on page 129.
2. Determine all Gateway instances that use the same `CTG_MASTER_RRMNAME` value. All Gateways with the same value are in the same Gateway group.
3. Remove the `CTG_RRMNAME` and `CTG_MASTER_RRMNAME` parameters from the STDENV file for each Gateway within the group.
4. Specify the same `APPLID` qualifier for each Gateway daemon which is to be a member of a Gateway in the group. It must be unique within the sysplex for this Gateway group. A Gateway daemon group is identified by the `APPLIDQUALIFIER` parameter. See Configuring identification using APPLID for a general explanation of `APPLID` and `APPLIDQUALIFIER`. See EXCI server connections for details on setting `APPLID` and `APPLIDQUALIFIER` when XA support is enabled.
5. You must permit access for the Gateway daemon USERID to one of the following RACF facilities when XA support is enabled:

   ALTER access to the MVSADMIN.RRS.COMMANDS.** facility.

   ALTER access to the MVSADMIN.RRS.COMMANDS.gname.sysname facility.

gname
is the logging group name and corresponds to the logging group in the RRS administrative panels in ISPF. Set **gname** to the value for the sysplex where the Gateway is running.

sysname
is the system name. Set **sysname** to the value for the LPAR where the Gateway is running

If you give ALTER access to MVSADMIN.RRS.COMMANDS.** the Gateway daemon is permitted to perform recovery operations for transactions associated with any system name or logging group. This option requires less administration but does not provide granularity of control.

If you give ALTER access to MVSADMIN.RRS.COMMANDS.gname.sysname the Gateway daemon is permitted to perform recovery operations for transactions associated with the specified system name or logging group. This option allows for greater granularity of control but requires a greater amount of administration.

Related reference:

"Using multiple releases of CICS TG" on page 15
You can run multiple versions of the Gateway in the same LPAR. To enable this when using XA support, the CTGRRMS services must be running, and must be at the level of the latest release of CICS Transaction Gateway installed on the LPAR.

Upgrading from Version 7 Release 0

This information details changes that you need to consider when upgrading to the current release from V7.0.

Configuration files

CICS Transaction Gateway configurations that worked with previous releases might not work after 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.

The Configuration Tool is removed from this version of CICS Transaction Gateway for z/OS. To assist in configuring CICS Transaction Gateway additional scenario configuration files are provided. For more information, see Scenarios.

Java Version 7

Ensure the PATH environment variable in the STDENV file used during CICS TG initialization contains the location of the IBM Java 7 runtime environment.

Removal of Gateway daemon resource parameters

The **`uowvalidation`** parameter is no longer supported. This means that LUWs tokens are validated so that they can be used only from the client application connection from which the LUW was started.

If the **`uowvalidation`** parameter is specified in the configuration file, the Gateway daemon fails to start. You must remove the **`uowvalidation`** parameter from the configuration file.

If any of your applications rely on the **`uowvalidation`** parameter being turned off, the application receives the ECI_ERR_LUW_TOKEN error when trying to use the LUW token on a connection, to the Gateway daemon, that did not start the LUW.

Upgrading a statistics API port definition

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.

Message_Qualifier API removal

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.

Using the JEE interfaces in nonmanaged mode

The JAR file cicsj2ee.jar file is renamed to cicsjee.jar.

Supported characters in server names

Server names must now use characters from the supported character list (see "Server name" on page 30) 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.

Removal of ciphersuites=128bitonly parameter

Use of the **`ciphersuites=128bitonly`** parameter is deprecated.

SSL keyring settings moved

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: **`esmkeyring`**, **`hwcrypt`**, **`keyring`**, **`keyringpw`**, and **`keyringpwscrambled`**. The **`esmkeyring`** and **`hwcrypt`** parameters now take a parameter value when defined in the PRODUCT section.

TLS cipher suites

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

Upgrading from Version 6 and earlier

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

For more information, seeCICS Transaction Gateway Information Center Library.

Chapter 5. Security

Security mechanisms include link, bind and user security on connections, SSL client authentication, SSL server authentication, and identity propagation.

Security considerations

CICS Transaction Gateway can perform authentication and authorization checks at different points during the processing of requests

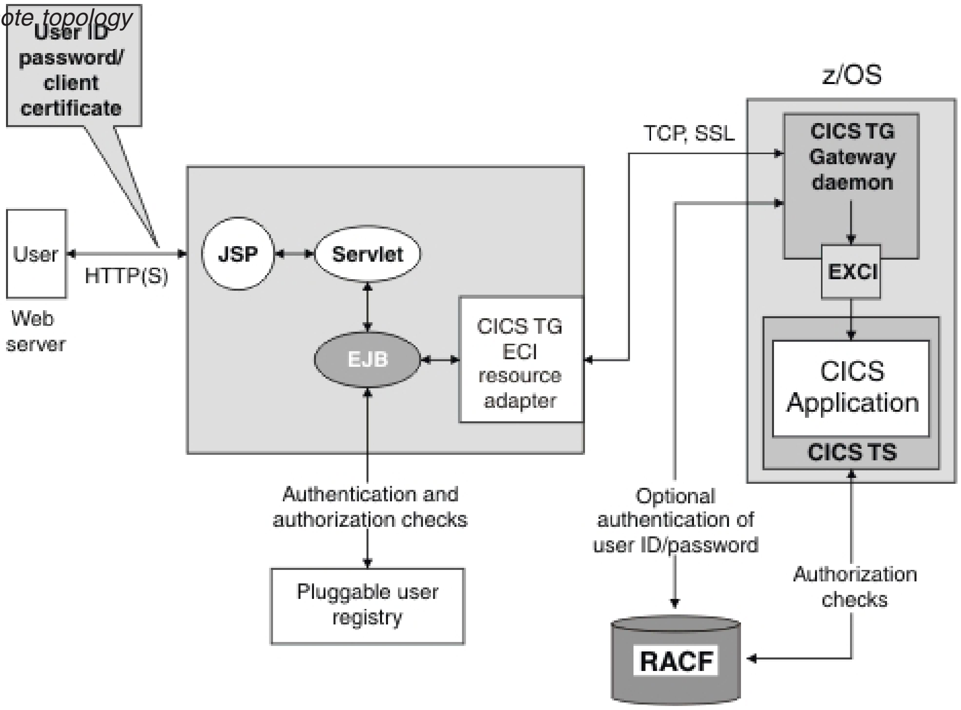
Authentication verifies that the user is who they say they are. Depending on topology, authentication can be based on the user ID passed with the ECI request, an SSL client certificate, or a distributed identity (identity propagation).

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 remote mode topology

The following figure shows the locations in a *remote mode* topology where the system performs authentication and authorization. In this topology, WebSphere Application Server is running on Windows and CICS Transaction Gateway is running on z/OS. The EJB application in WebSphere uses the ECI resource adapter and the Gateway daemon to access the CICS COMMAREA application.

Figure 5. Security in a remote topology



User ID password/ client certificate
z/OS
TCP, SSL
CICS TG Gateway daemon
User
HTTP(S)
Web server
JSP
Servlet
EJB
CICS TG ECI resource adapter
EXCI
CICS Application
CICS TS
Authentication and authorization checks
Pluggable user registry
Optional authentication of user ID/password
Authorization checks
RACF

The following *authentication* options are available in this topology:

- User authentication by CICS Transaction Gateway. The user ID can be passed to CICS without a password.
- Identity propagation. This is a unified security solution that enables additional user auditing and authorization by passing a distributed identity to CICS instead of a user ID and password.
- SSL client authentication. A trust relationship is established between WebSphere Application Server and the Gateway daemon so that the application server can be trusted to pass the user ID on an ECI request to CICS.

The following *authorization* options are available in this topology:

- Component-managed sign-on. With this option, security credentials are propagated to CICS by 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. This provides an additional check on whether the link user ID is authorized to access the CICS resource.
- MRO bind security. This prevents unauthorized attached MRO regions from starting transactions in a CICS server, and determines whether or not a particular CICS Transaction Gateway can connect (bind) to a particular CICS server.
- Link security. This Ensures that the link user ID used for authorization checks in CICS is the user ID associated with the started task of the Gateway daemon.
- Surrogate security. This authorizes the user ID associated with the CICS Transaction Gateway started task to switch the security context of an EXCI request to the user ID that was passed to CICS.

The following *data integrity and confidentiality* options are available in this topology:

- RACF keyring support. With this option SSL key stores are stored in RACF.
- System z hardware cryptographic support. SSL handshakes can be offloaded to hardware to reduce the CPU load due to handshakes and encryption.
- SSL cipher suite selection. This allows only certain algorithms and strengths of ciphers to be used for SSL connections

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.

EXCI connection security

EXCI connections enforce link, bind and user security. Link security restricts the resources that can be accessed over a connection to a CICS server, bind security prevents an unauthorized client system from connecting to CICS, and user security restricts the CICS resources that can be accessed by a user.

Link security

By default, the link user ID that CICS uses for these security checks is the user ID under which the Gateway daemon runs; to override this, specify a USERID parameter in the SESSIONS definition.

Bind security

The client application is treated in the same way as a CICS server for MRO logon and connect (bind-time) security checking; when the client connects, the CICS interregion communication program (IRP) performs logon and bind-time security checks against the user ID under which the client is running.

User security

A number of settings and security checks ensure validation of user IDs and passwords.

- The user ID and password or password phrase coded on the ECI request object can be validated in the CICS Transaction Gateway through RACF for every EXCI call. This is controlled through the setting of the AUTH_USERID_PASSWORD environment variable. For more information, see the *CICS Transaction Gateway: z/OS Administration*.
- The ECI user ID can then be subjected to an optional surrogate security check, if the flowed user ID is different from the user ID in the EXCI address space. This option is specified using the SURROGCHK parameter in the EXCI options module DFHXCOPT, for more information, see the CICS Transaction Server Information Center. Note that any password supplied on an ECI request is not flowed on to CICS from CICS Transaction Gateway.
- The flowed user ID is subject to CICS authorization checks, for more details, see the *CICS Transaction Server for z/OS RACF Security Guide*.

See also “Configuring for client certificate mapping” on page 143.

A user ID can also be obtained from a mapping of an SSL client certificate. For more information, see “User authentication using SSL client certificates” on page 49.

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 Information Center.

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 “Identity propagation” on page 51.

A user ID can also be obtained from a mapping of an SSL client certificate.

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.

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:

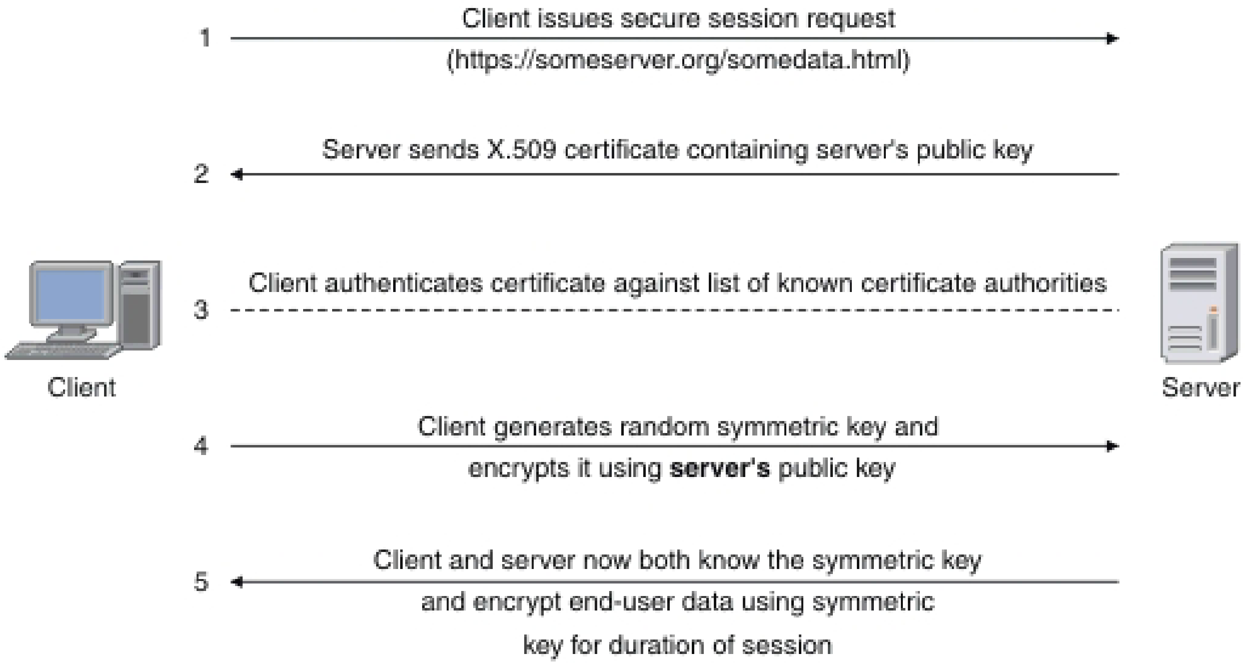

1
Client issues secure session request
(https://someserver.org/somedata.html)
2
Server sends X.509 certificate containing server's public key
3
Client authenticates certificate against list of known certificate authorities
Client
Server
4
Client generates random symmetric key and
encrypts it using **server's** public key
5
Client and server now both know the symmetric key
and encrypt end-user data using symmetric
key for duration of session

Figure 6. SSL handshake

1. The client sends a request to the server for a secure session. The server responds by sending its X.509 digital certificate to the client.
2. The client receives the server's X.509 digital certificate.
3. The client authenticates the server, using a list of known certificate authorities.
4. The client generates a random symmetric key and encrypts it using server's public key.
5. The client and server now both know the symmetric key and can use the SSL encryption process to encrypt and decrypt the information contained in the client request and the server response.

CICS Transaction Gateway supports the JSSE implementation of SSL. JSSE as supplied with the Java SDK is the only supported option. For more information, see Chapter 5, “Security,” on page 41.

Authentication

During server authentication, a connection is only established if the client trusts the server based on the information presented by the server to the client in its certificate.

During client authentication (if activated) the client sends its certificate information to the server. A connection is then only established if the client trusts the server *and* the server trusts the client, based on the information exchanged in both certificates.

Transport Layer Security (TLS):

Network connections between a JEE client and CICS can be secured by the Secure Sockets Layer (SSL) protocol, or the Transport Layer Security (TLS) protocol.

TLS is an industry-standard SSL protocol. The TLS specification is documented in RFC2246; for more information, see . `http://www.rfc-editor.org/rfcsearch.html`

All references to SSL in this information center also apply to TLS. Connections that require encryption automatically use the TLS protocol, unless the client specifically requests SSL. For more information on configuring CICS Transaction Gateway to use network security, see “Configuring SSL” on page 132.

No special configuration or upgrade tasks are required for using TLS, when compared with SSL.

Encryption:

Cryptography is the scientific discipline for the study and development of ciphers, in particular, encryption and decryption algorithms. These cryptographic procedures are the essential components that enable secure communication to take place across networks that are not secure. SSL encryption uses both symmetric and asymmetric keys.

Symmetric (secret) key

Secret key cryptography means that the sender and receiver share the same (symmetric) key, which is used to encrypt and decrypt the data.

The secret key encryption and decryption process is often used to provide privacy for high-volume data transmissions.

Asymmetric (public/private) key

Public/private key cryptography uses an asymmetric algorithm. The private key is known only by its owner and is never disclosed. The corresponding public key can be known by anyone. The public key is derived from the private key, but it cannot be used to deduce the private key. Either key of the pair can be used to encrypt a message, but decryption is only possible with the other key.

Digital signatures, certificates and key rings:

SSL uses digital signatures and digital certificates for establishing a trusted relationship between a sender and a receiver of information sent over a network connection.

Digital signature

A digital signature is a unique, mathematically computed, signature that demonstrates the authenticity of a transmission.

Digital certificate

A digital certificate allows unique identification. It is essentially an electronic ID card, issued by a trusted third party known as a certificate authority. Digital certificates form part of the ISO authentication framework, also known as the X.509 protocol. This framework provides for authentication across networks. A digital certificate serves two purposes: it establishes the owner's identity and it makes the owner's public key available.

A digital certificate contains the following information:

- public key of the person being certified
- name and address of the person being certified, also known as the Distinguished Name (DN)
- digital signature of the certificate authority
- issue date
- expiry date

If you send your digital certificate, containing your public key, to someone else, your private key prevents that person from misusing your digital certificate and posing as you.

A digital certificate alone is not proof of an identity; it allows verification of the owner's identity, by providing the public key needed to check the owner's digital signature. Therefore, the digital certificate owner must protect the private key that belongs with the public key in the digital certificate. If the private key is stolen, anyone could pose as the legitimate owner of the digital certificate.

Certificate authority (CA)

A digital certificate is issued by a CA and has an expiry date. When requesting a digital certificate, you supply your distinguished name. The digitally signed certificate includes your distinguished name and the distinguished name of the CA. This allows verification of the CA.

To communicate securely, the receiver must trust the CA that issued the certificate that the sender is using. Therefore, when a sender signs a message, the receiver must have the corresponding CA's signer certificate and public key designated as a trusted root key. Your Web browser has a default list of signer certificates for trusted CAs. If you want to trust certificates from another CA, you must receive a certificate from that CA and designate it as a trusted root key.

Key ring

A key ring is a file that contains the digital certificates, public keys, private keys, and trusted root keys used by a network communications security protocol such as SSL. Each certificate consists of a public key and a private key. A root certificate contains a trusted root key.

SSL requires access to key rings for the establishment of secure connections. The key rings used by the Java Secure Socket Extension (JSSE) implementation of SSL are known as *KeyStores*.

For information on how to create key rings, see “Configuring SSL” on page 132.

Cipher suites:

A cipher suite is a set of ciphers (encryption algorithms) used for encrypting sensitive information. SSL uses cipher suites to ensure security and integrity of information transmitted over a network connection. Different cipher suites provide different levels of encryption.

To allow users to select the level of security that suits their needs, and to enable communication with others who might have different needs, SSL defines cipher suites, or sets of ciphers. When an SSL connection is established, the client and server exchange information about which cipher suites they have in common. They then communicate using the common cipher suite that offers the highest level of security. If they do not have a cipher suite in common, secure communication is not possible.

There are many different algorithms that can be used for encrypting data, and for computing the message authentication code. Some provide the highest levels of security, but require a large amount of computation for encryption and decryption; others are less secure, but provide rapid encryption and decryption. The length of the key used for encryption affects the level of security; the longer the key, the more secure the data.

The individual ciphers that can be used by CICS are dependent on the CICS Transaction Server ENCRYPTION parameter. This is a system initialization parameter which can be set for weak, medium or strong encryption.

User authentication using SSL client certificates

You can optionally use client certificates with SSL to allow the server to authenticate the client during the SSL handshake.

A client certificate can be used with or without another authentication mechanism such as a user ID and password. When a client certificate has been authenticated it can be made available on each ECI request, and can be used by the Gateway daemon security exit to authorize the request. This is achieved by mapping the certificate to a RACF user ID.

To obtain the client certificate, client authentication must be enabled on the SSL protocol handler in the Gateway daemon. To run the CICS transaction under the RACF user ID which has been mapped to the client certificate, ensure that the CICS connection has been defined with **Attachsec** set to `Identify`.

To map a certificate to a RACF user ID, you must first associate the certificate with a RACF user ID, using one of the following procedures:

- By using the RACF command RACDCERT. If you use this procedure, the client certificates are stored in the RACF database, and a user ID is associated with them. This is an excellent way to create a one-to-one mapping between client certificates and user IDs, but it does not scale well if large numbers of certificates must be mapped to a small number of user IDs. For more information, see “Associating a client certificate with a RACF user ID.”
- By using RACF certificate name filtering. If you use this procedure, rules are applied to allow multiple certificates to be assigned to a single user ID with one profile. For more information, see “RACF certificate name filtering” on page 50.

For more information on certificate mapping, see the IBM Redpaper *J2C Security on z/OS* at:

`http://www.redbooks.ibm.com/redpapers/pdfs/redp4202.pdf`

Associating a client certificate with a RACF user ID

You can create a profile that associates a client certificate with a specified user ID. The profile can then be used for translating a certificate to a user ID, without the need for a password.

1. Perform the actions described in “Configuring for client certificate mapping” on page 143.

2. Copy the certificate that you wish to process into an MVS™ sequential file. The file must have a variable length, blocked records (**RECFM=VB**) and be accessible from TSO.
3. Run the **RACDCERT** command in TSO. The syntax of **RACDCERT** is:

   ```
   RACDCERT ADD('datasetname') TRUST [ ID(userid) ]
   ```

   where:

   datasetname
   is the name of the data set containing the client certificate.

   userid is the user ID to be associated with the certificate. This parameter is optional. If omitted, the certificate is associated with the user issuing the **RACDCERT** command.

When you issue the **RACDCERT** command, RACF creates a profile in the **DIGTCERT** class. This profile associates the certificate with the user ID. You can then use the profile to translate a certificate to a user ID without giving a password.

For further information on the **RACDCERT** command, including the format of data allowed in the downloaded certificate data set, see *z/OS Security Server (RACF) Command Language Reference*.

RACF certificate name filtering

With certificate name filtering, distinct client certificates do not have to be defined to RACF for every individual user.

The association between one or more certificates and a RACF user ID is achieved by defining a filter rule that matches the distinguished name of the certificate owner or issuer (CA). A sample filter rule might look like this:

```
RACDCERT ID(DEPT3USR) MAP SDNFILTER(OU=DEPT1.OU=DEPT2.O=IBM.L=LOC.SP=NY.C=US)
```

This sample filter rule would associate user ID DEPT3USR with all certificates when the distinguished name of the certificate owner contains the organizational unit DEPT1 and DEPT2, the organization IBM, the locality LOC, the state/province NY and the country US.

Determining the RACF user ID associated with a certificate

The RACFUserid class can be used in conjunction with the CICS Transaction Gateway security exit to map an ECI request to a RACF user ID, based on the distinguished name in the SSL client certificate. The class has the following methods:

- **setCertificate(byte[] clientCertificateData);**
- **getRACFUserid();**

Create a **RACFUserid** as follows:

```
RACFUserid myUseridObject = new RACFUserid(myCertificate);
```

This creates an object and automatically populates it with certificate data, without needing to call the **setCertificate(byte[] clientCertificateData);** method. When the object has been created, the **getRACFUserid()** method can be used to make a native RACF call to determine the user ID associated with the certificate data. If successful, it returns a string containing the user ID.

The **SSLServerCompression.java** class in the <install_path>/samples/java/com/ibm/ctg/samples/security subdirectory shows an example of how to use the **RACFUserid** class.

For more information on the **com.ibm.ctg.util.RACFUserid** class, see the information about Class RACFUserid in the Javadoc.

Related information:

CICS Transaction Gateway security classes

Identity propagation

CICS Transaction Gateway can pass user security identity information (a distributed identity) from a JEE client in WebSphere Application Server across the network to CICS Transaction Server for z/OS. The security identity of the user is preserved for use during CICS authorization and for subsequent accountability and trace purposes.

Identity propagation provides a way of authorizing requests by associating security information in WebSphere Application Server with security information in CICS Transaction Server for z/OS.

CICS Transaction Gateway supports identity propagation for JEE client requests from WebSphere Application Server to CICS Transaction Server for z/OS. Identity propagation is supported when using a CICS Transaction Gateway ECI resource adapter and an IPIC connection to CICS.

Distributed identities can be tracked using the request monitoring exits, see "Request monitoring exits" on page 307 for more information.

Benefits of using identity propagation

Identity propagation provides end-to-end security and consistent accountability, when applications in WebSphere Application Server are connected to CICS.

Identity propagation provides the following benefits:

- An end-to-end solution for security when connecting WebSphere Application Server to CICS Transaction Server for z/OS.
- A unified mechanism for authentication using security information stored in different formats on different user registries such as IBM Tivoli Directory Server or WebSphere Portal. For more information, see the documentation for WebSphere Application Server.
- "Single sign-on" authentication of users in WebSphere Application Server before they are authorized in CICS Transaction Server for z/OS.
- Consistent accountability.

Configurations that support identity propagation

A range of products and network topologies support identity propagation.

Products that support identity propagation

The following IBM products support identity propagation:

- All versions of WebSphere Application Server supported by CICS Transaction Gateway. For more information, see "JEE application servers" on page 10.

- Any user registry that is supported by WebSphere Application Server. For more information, see the documentation for WebSphere Application Server.
- CICS Transaction Server for z/OS Version 4.1 (with APAR PK83741 and APAR PK95579), or later. For more information, see the CICS Transaction Server for z/OS information center.
- IBM z/OS Version 1.11 or later.
- IBM RACF Security Server for z/OS Version 5 or later. For more information, see *Introduction to CICS Security with RACF* in the CICS Transaction Server for z/OS information center.

Network topology for using identity propagation

Identity propagation is supported when connecting to CICS using an IPIC connection. A client authenticated SSL connection is required unless CICS Transaction Gateway and CICS Transaction Server are on z/OS and on the same sysplex

For more information about the topologies that are supported by CICS Transaction Gateway, see “Deployment topologies” on page 2.

The following example shows identity propagation in a remote mode topology:

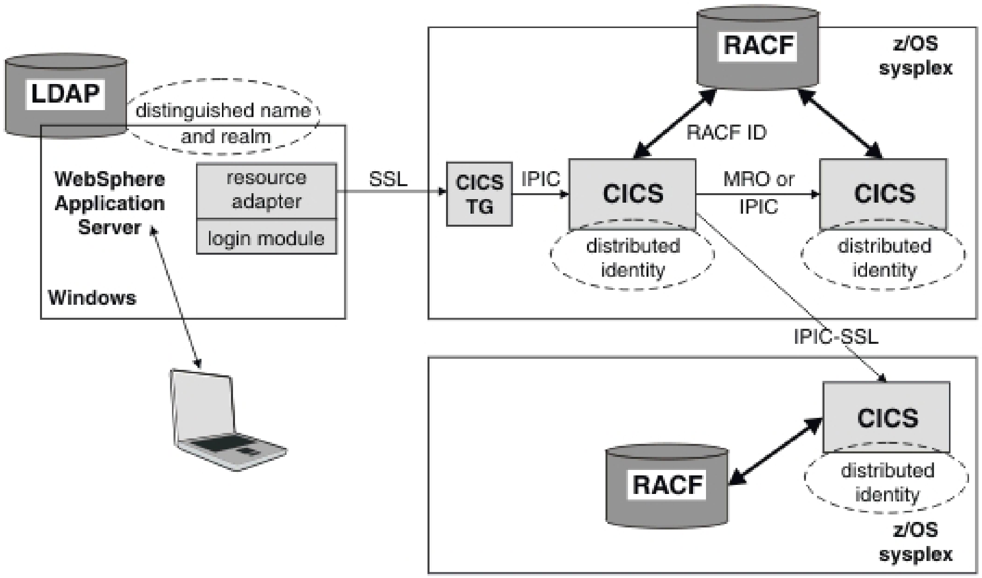


Figure 7. Example of identity propagation in a remote mode topology

The user security information consists of a distinguished name and a realm name. The distinguished name uniquely identifies an entry within a user registry. The realm name represents a named collection of users and groups that can be used in a specific security context.

When the user has been authenticated in WebSphere Application Server, the security information is passed unchanged as a *distributed identity* to CICS. The distributed identity is mapped to a RACF user ID, which is used for authorization by CICS.

Precedence of distributed identities over asserted user IDs

A distributed identity takes precedence over user IDs that have been asserted directly using other mechanisms.

The identity used by CICS Transaction Server depends on whether a distributed identity has been specified and whether a valid mapping exists:

Distributed identity supplied and valid RACF mapping exists	Distributed identity supplied but valid RACF mapping does not exist	Distributed identity not supplied
The distributed identity is used and any specified user ID is ignored.	If a user ID is specified and is valid, that user ID is used.	If a user ID is specified and is valid, that user ID is used.

If a user is not authenticated by the WebSphere Application Server user registry, a distributed identity is not used even if the CICS Transaction Gateway identity propagation login module is enabled. In this situation, if a user ID has been specified in the connection factory or application, that user ID is used.

Chapter 6. Performance

The performance of individual components, including CICS Transaction Gateway, can affect overall system performance.

Related reference:

"List of statistics" on page 319
These statistics are available from the CICS Transaction Gateway.

Related information:

"Displaying statistics" on page 315
You can use MVS system commands to display statistical information about the CICS Transaction Gateway, or obtain statistics using either the C or Java Statistics API interface.

Performance indicators and factors

The performance of CICS Transaction Gateway can be measured to understand the factors that affect performance, and to use the information provided to optimize that performance.

Performance indicators

Performance indicators 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.

EXCI or IPIC

Performance over EXCI is generally better than IPIC. However, IPIC performance can be greatly improved if a zAAP is deployed which will significantly reduce the CPU cost per transaction. EXCI is limited to 32K payloads using a COMMAREA, whereas IPIC payloads can be as large as 50MB using channels and containers.

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:
- Performance monitoring tools such as RMF (Resource Management Facility), request monitoring exits, and IBM Tivoli OMEGAMON XE for CICS (on z/OS)
- Statistics for monitoring and managing system resources.
- For more information see “Statistics and monitoring” on page 6.

Benefits of using a 64-bit Gateway

If you run your Gateway with large numbers of clients and large container payloads, for example, 2 MB or more, you should consider using a 64-bit z/OS Gateway.

For the Gateway daemon running under the 64-bit SDK for z/OS. the memory requirements vary based on environmental factors or the characteristics of the workload. This topic describes a configuration which could not be achieved with a Gateway daemon using the 31-bit SDK for z/OS.

In this example, a workload of 250 concurrent clients each sending ECI requests with 2 MB containers to CICS TS, requires a Gateway daemon defined with 250 connection managers, 250 worker threads, and 250 IPIC send sessions. To avoid running out of memory you would require a maximum heap size (`-Xmx`), of at least 1536 MB. Better performance is obtained with a maximum heap size of 2048 MB, to avoid excessive garbage collection. A Gateway daemon with a heap size of 2048 MB is only possible with the 64-bit SDK for z/OS.

The maximum heap size directly affects the value that must be used for MEMLIMIT, and the Gateway daemon threads still utilize 31-bit storage. In this example, the REGION size was set to 300 MB, tuned by reviewing the **`SE_CELOAL`** statistic, and **`MEMLIMIT`** set to 3.3 GB.

To tune the heap size, you should review the **`SE_CHEAPGCMIN`** statistic as a percentage of **`SE_SHEAPMAX`** to ensure the value is in the range 40-70%. If the percentage goes above 70% you can increase the maximum heap size. If the percentage is below 40% then you can reduce the maximum heap size.

Increasing the ECI payload size to 4 MB containers, the heap size needs to be increased to avoid excessive garbage collection. In the example of 250 clients, `-Xmx` was set to 2048 MB. Increasing the heap size requires **MEMLIMIT** to be increased. In this example, 3.73 GB was used, so **MEMLIMIT** was set to 4 GB.

Gateway daemon performance improvements may also be gained using the Java directive `-Xcompressedrefs` with 64-bit SDK for z/OS. The compressed references feature was added to the IBM Developer Kit for Java 6, 64-bit edition, J9 Java virtual machine (JVM) and Just-in-Time (JIT) compiler to provide relief for memory footprint growth incurred when migrating from a 31-bit JVM to a 64-bit JVM.

Table 1 contains some example scenarios displaying the benefits of using a 64-bit gateway with increasing channel sizes: All scenarios use 200 clients, with 250 connection managers, 250 worker threads and 250 IPIC Send Sessions defined in the configuration file.

Table 2. Scenarios using 64 bit Gateway. Benefits of using a 64-bit gateway with increasing channel sizes.

Channel Size	MEM LIMIT	REGION	SE_CELOAL	SE_C31 MAX	Max Heap	SE_CHEAPGC MIN	SE_S HEAP MAX	Heap Occupancy
1 MB	4 GB	300 MB	262 MB	1688 MB	2048 MB	320010016	2147483648	15%
2 MB	4 GB	300 MB	262 MB	1688 MB	2048 MB	315898512	2147483648	15%
3 MB	4 GB	300 MB	262 MB	1688 MB	2048 MB	492042016	2147483648	23%
4 MB	4 GB	300 MB	262 MB	1688 MB	2048 MB	451839424	2147483648	21%

For optimum performance you should aim for a heap occupancy of between 30% and 70%.

In a similar scenario with a 31-bit Gateway, only a 1 MB channel with 200 clients is possible. 2 MB channels caused the Gateway daemon to run out of memory. See Table 2.

Table 3. Scenarios using 31 bit Gateway. A scenario using 31-bit gateway to compare with 64-bit gateway.

Channel Size	MEM LIMIT	REGION	SE_CELOAL	SE_C31 MAX	Max Heap	SE_CHEAPGC MIN	SE_S HEAP MAX	Heap Occupancy
1 MB	n/a	800 MB	800 MB	1688 MB	500 MB	195826792	524288000	37%

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 84 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 84 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 to destroy, JavaGateway objects.

Worker threads

Worker threads handle outbound connections between CICS Transaction Gateway and your CICS server. The design of your applications, and the workload that you need to support, affects the number of worker threads you need: the longer your CICS transactions remain in process, the more worker threads you need to maintain a given transaction rate.

If the value specified for **Initial number of worker threads** is too high, CICS Transaction Gateway uses resources to manage threads that it does not need.

If the value is too low, CICS Transaction Gateway uses resources to search for available worker threads.

See "Initial number of worker threads" on page 84 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.

An MVS address space cannot open more than the maximum number of EXCI pipes set by CICS. This means that you cannot improve performance by specifying a value greater than that set by CICS for "Maximum number of worker threads" on page 85. However, if your CICS transactions remain in process for a long time, all of the EXCI connections could become busy, leaving no worker threads available.

You can increase performance by starting multiple gateways; see "Starting multiple CICS Transaction Gateways" on page 254.

Display TCP/IP host names

Selecting this option might cause severe performance reduction on some systems. See "Display TCP/IP hostnames" on page 87.

Timeout values

It is unlikely that you can improve performance by changing the default timeout values. However, you might need to change them for particular applications. See "Configuring Gateway daemon settings" on page 83 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 87.

Local mode

Design your system configuration to use local mode communication between components wherever possible.

In local mode, WebSphere Application Server and the CICS Transaction Gateway are installed on the same z/OS image. Calls to the CICS Transaction Gateway are made using a local protocol. This allows calls from the CICS resource adapters to go directly to the CICS Transaction Gateway JNI layer.

Remote mode

In remote mode, all calls to the CICS Transaction Gateway are made using a standard network protocol such as TCP or SSL. This will happen even if WebSphere Application Server and the CICS Transaction Gateway are installed on the same machine. If your configuration has components running in remote mode, additional service provider daemons will be running to process and forward requests between components. These extra demands on the system can have an adverse effect on performance.

Related reference:

"List of statistics" on page 319
These statistics are available from the CICS Transaction Gateway.

Related information:

"Displaying statistics" on page 315
You can use MVS system commands 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 "Gateway daemon resources" on page 83 for information on setting configuration parameters.

You can also specify these limits when you start CICS Transaction Gateway. For more information see "Starting from a command line" on page 254.

Consider these thread limits when setting the number of connection manager and worker threads:

System-wide limit of the maximum number of threads	Process limit of the number of threads
This might be restricted by the total number of MVS Task Control Blocks (one is created for each UNIX System Services thread.)	This limit is governed by the UNIX System Services parameters MAXTHREADS and MAXTHREADTASKS.

The total number of threads in use by the Gateway daemon can be displayed using the MVS system command /D OMVS,L,PID=nnnn , where nnnn is the process ID of the JVM running the Gateway daemon, as displayed using the SDSF PS menu option. You can also determine what values are set for the UNIX System

Services parameters MAXTHREADS and MAXTHREADSTASK by examining the appropriate BPXPRMxx member of SYS1.PARMLIB. For more information about these parameters, see the *z/OS UNIX System Services Planning*.

The threading model is illustrated in the following figure:

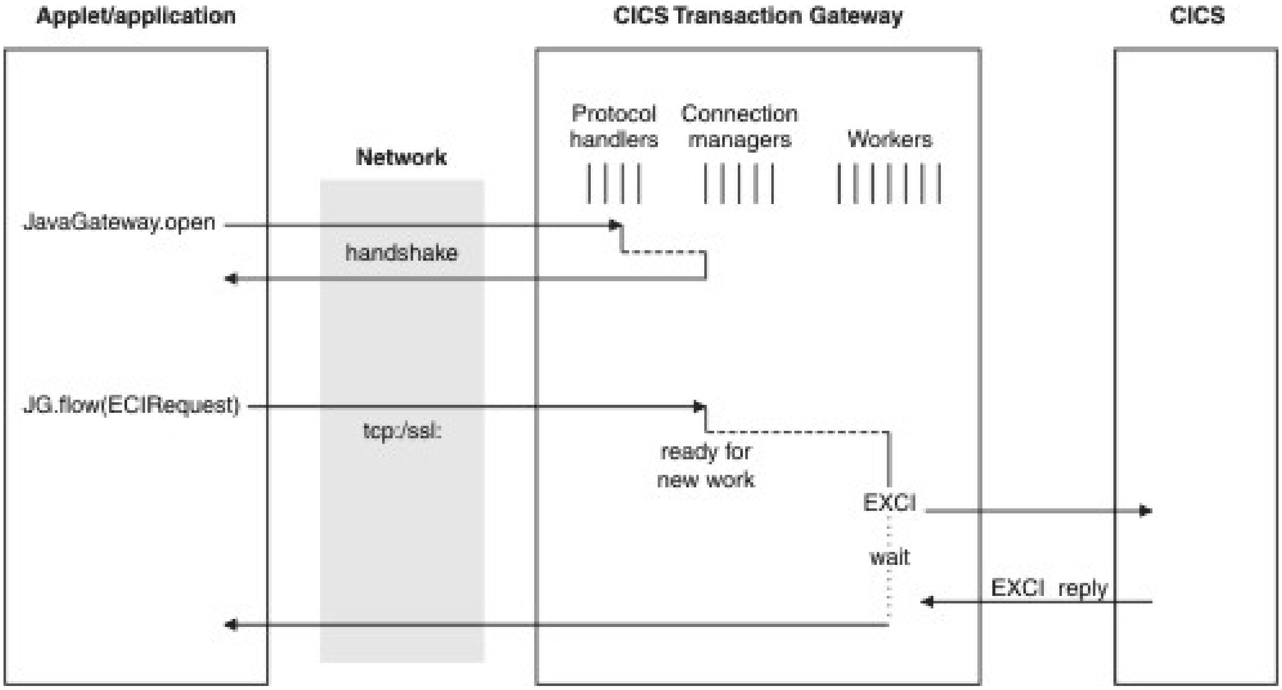


Figure 8. CICS Transaction Gateway Threading model for TCP/IP and SSL protocols using a persistent socket

For information on how EXCI pipes constrain the maximum number of threads, see “Maximum number of worker threads” on page 85.

Related reference:

“List of statistics” on page 319
These statistics are available from the CICS Transaction Gateway.

Related information:

“Displaying statistics” on page 315
You can use MVS system commands to display statistical information about the CICS Transaction Gateway, or obtain statistics using either the C or Java Statistics API interface.

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 available 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_SELIM

- SE_CELOAL
- SE_C31MAX
- SE_SHEAPMAX
- WT_SMAX
- WT_CCURR

Gateway daemon storage requirements, with 31-bit Java

The numbers quoted are based upon observations with 31-bit SDK for z/OS, Java Technology Edition, V7 (SR1), running with the default operating system thread stack size 256K.

Three factors affect the 31-bit virtual storage requirements of a Gateway daemon address space:

- The maximum JVM heap size, SE_SHEAPMAX. The default value is 128 MB; this is sufficient for the default values of the thread parameters (100 connection manager, 100 worker threads). Larger heaps must be considered where IPIC connections are used, and the workload includes larger channel payloads. You should allow 300KB for each IPIC SENDSESSION.
- The stack storage used by the Gateway daemon threads; plan for 128 KB per Connection Manager thread, and 1200 KB per worker thread. If EXCI is disabled, allow 650 KB per Worker thread.
- The storage footprint required to run the Gateway daemon; this is approximately 70 MB per Gateway but it varies if you use additional functions such as XA, WLM health reporting, SMF recording, and SSL.

If a Gateway daemon attempts to create a new thread and the additional stack storage needed requires more virtual storage than is available to the address space, an out of memory condition is produced. This condition is likely to cause the Gateway daemon to shut down. Use the following calculation to avoid these out of memory conditions; it estimates the potential maximum virtual storage requirements of the Gateway daemon. The calculation gives a result in kilobytes.

When EXCI is enabled (by default):

```
 70000 + (SE_SHEAPMAX / 1024) + (128 * CM_SMAX ) + (1200 * WT_SMAX )
```

When EXCI is disabled (environment variable CTG_EXCI_INIT=NO):

```
70000 + (SE_SHEAPMAX / 1024) + (128 * CM_SMAX ) + (650 * WT_SMAX )
```

When an application requires the use of very large numbers of connection manager and worker threads, and demands on memory usage are high, consider splitting the application across a number of LPARs, or across several machines. Servicing very large numbers of Java threads within a single JVM imposes significant performance overheads.

If you increase the JVM maximum heap size, ensure that you also increase the address space limit by the same amount. Increasing the JVM maximum heap size reduces the amount of memory that is available for the creation of worker threads and connection manager threads, and an out of memory condition might occur unless you have increased the address space limit by the same amount.

A small saving per thread can be made by using the minimum operating system thread stack size of 128k (-j-Xmso=128K via CTGSTART_OPTS).

Gateway daemon storage requirements, with 64-bit Java

The Gateway daemon memory requirements when running under 64-bit SDK for z/OS, Java Technology Edition, V7 are very different from 31-bit. For more information, see "Benefits of using a 64-bit Gateway" on page 56.

31-bit storage tuning

This information applies to 31-bit storage tuning, when you run the Gateway daemon for z/OS with 31- or 64-bit Java. However, this is most relevant if you are use 31-bit Java. The 31-bit virtual storage used by a Gateway daemon address space, that is reported by the SE_CELOAL statistic, cannot exceed the available 31-bit storage. Precisely how much 31-bit storage is available, and how to tune the 31-bit storage for a Gateway daemon system depends upon whether a specific limit is imposed, for example; **REGION=***500M*, or if the Gateway daemon is set to use the maximum available, for example; **REGION=***0M*.

Running with a 31-bit storage constraint

Running a Gateway daemon with an upper limit on 31-bit storage set, that is with a non-zero **REGION** parameter, the amount of 31-bit storage that is currently used by the Gateway daemon, statistic SE_CELOAL, must not approach the available limit, SE_SELIM. When these two statistics become too near in value, OutOfMemoryExceptions might occur that cause the Gateway daemon to fail.

In this case, statistic SE_C31MAX provides a dynamic indication of the limit of the available 31-bit storage. Therefore, you can use SE_C31MAX as a guide for increasing REGION size, and it represents the largest size that REGION can achieve. However, SE_C31MAX is a dynamic value, and is at a lowest value during peak workload. You should adopt this lowest value as guidance if your goal is to maximize the 31-bit storage usage in a single Gateway daemon.

Typically, when REGION is set to a non-zero value you will observe the following relation between the SE resource group statistics:

```
SE_CELOAL < SE_SELIM < SE_C31MAX
```

These three statistics identify:
- How close the Gateway daemon is to the configured 31-bit storage constraint, SE_CELOAL vs SE_SELIM .
- By how much the Gateway daemon 31-bit storage constraint can be increased, SE_SELIM vs SE_C31MAX.

Running without a 31-bit storage constraint

Running a Gateway daemon with no upper limit on 31-bit storage set, for example; **REGION**=*0M*, the amount of 31-bit storage that is currently used by the Gateway daemon, statistic SE_CELOAL, must not approach the limit of 31-bit storage, SE_C31MAX. When these two statistics become too near in value, OutOfMemoryExceptions might occur that cause the Gateway daemon to fail.

In this case, statistic SE_C31MAX provides the only accurate upper limit of the amount of 31-bit storage that is currently used by the Gateway daemon, statistic SE_CELOAL. This is because when REGION=0M is used, the SE_SELIM value does not account for ELSQA storage, which varies dynamically in relation to workload.

Typically, when REGION is set to zero, REGION=0M, then you can observe the following relation between SE resource group statistics:

```
SE_CELOAL < SE_C31MAX < SE_SELIM
```

Observing statistics SE_CELOAL and SE_C31MAX can help you to:

- Evaluate the potential for increasing numbers of thread or heap size in a particular Gateway daemon.
- Move to a configuration with a constraint on 31-bit virtual storage.

The lowest value of SE_C31MAX, observed at peak workload, represents the highest value which you might impose as a constraint on 31-bit virtual storage for a Gateway daemon. The maximum value of SE_CELOAL, observed at peak workload, represents the lowest value which you might impose as a constraint on virtual 31-bit storage for a Gateway daemon.

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. For more information, see “Configuring Java shared classes” on page 97.

“System environment statistics” on page 330 are available to show the following Java statistical information:

- Region storage usage on z/OS
- 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 Transaction Gateway 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.

Tuning JEE

Because of the overheads associated with XA transactions, the use of network and processor resources is higher when using the XA transactional support provided by cicseci.rar with the xasupport custom connection factory property set to `on`.

The decision on how to configure the resource adapter depends on the transaction support required by the JEE application. For more information see"Transaction management models" on page 170.

For transactions that use XA, set **`xasupport`** to `on` in the connection factory. For performance reasons, you should define a second connection factory with **`xasupport`** set to `off` for those transactions that do not use XA.

WebSphere Application Server connection pooling

If the Client application is running under WebSphere Application Server, connection pooling can be used in remote mode to reduce the overheads associated with establishing connections.

Configuring z/OS parameters

z/OS configuration parameters needed to run CICS Transaction Gateway

Unlike EXCI connections, IPIC provides the ability to use more than 250 worker threads in a single Gateway. To enable this function the Gateway daemon must be started with environment variable **`CTG_EXCI_INIT`**`=NO`. If **`CTG_EXCI_INIT`**`=NO` is not set, EXCI support is loaded at Gateway daemon startup which restricts the maximum number of threads to the maximum EXCI **`LOGONLIM`** of 250.

Region size

The JVM runs within the CICS Transaction Gateway address space. This means that, if your application requires CICS Transaction Gateway to create large numbers of threads, for example connection manager threads, you might need to increase the region size. You define the region size with the **REGION** parameter in the **EXEC** statement in the CICS Transaction Gateway start JCL. See "Region size considerations" on page 252 for more information on this parameter.

EXCI considerations

For EXCI workloads, CICS Transaction Gateway and CICS must be configured to ensure there are is not an EXCI pipe shortage.

A connection manager uses one worker thread for every request to run a transaction in CICS. A worker thread allocates one EXCI pipe (MRO session) to every CICS server (APPLID) with which it communicates. This can be limited to a single pipe using the setting CTG_PIPE_REUSE.

Configure your system as follows:

- Set the maximum number of connection manager threads (**`maxconnect`**) in the Gateway daemon to the same value as the maximum number of connections in all connection factories that connect to the Gateway daemon.

- Set the initial number of connection manager threads (**`initconnect`**) to the same value as the minimum number of managed connections in the connection pool in all connection factories that connect to the Gateway daemon.
- Set the maximum number of worker threads to a value that meets both the following criteria:
  - Less than or equal to the maximum number of connection manager threads.
  - Less than or equal to the EXCI LOGONLIM.
- Set the initial number of worker threads to the same value as the initial number of connection manager threads.
- Set the RECEIVECOUNT for the CICS SESSIONS definition to a value that meets both the following criteria:
  - No less than the value for EXCI LOGONLIM.
  - The same value as the number of worker threads in the Gateway daemons that can connect through this connection (if multiple Gateway daemons share a connection).

When EXCI connections are used by a Gateway daemon, the maximum number of worker threads is restricted to LOGONLIM. Because all CICS server connections share the same pool of worker threads, this restriction also affects any IPIC connections. If EXCI has been disabled using the CTG_EXCI_INIT parameter, this restriction does not apply.

EXCI connections

When you define your EXCI connections you can choose between *generic EXCI pipes* and *specific EXCI pipes*. Specific EXCI pipes give slightly better performance. See “CICS server connection definition” on page 122 for more information on selecting your EXCI connections.

EXCI resource shortage

An insufficient number of EXCI pipes causes intermittent resource shortage errors to be reported to the API. Requests receive ECI_ERR_RESOURCE_SHORTAGE errors. This can occur when any of the following events is happening.

- Several CICS servers are being used by each Gateway and the CTG_PIPE_REUSE=ALL model is in use.
- Large numbers of worker threads are being used.
- Requests receive EXCI return code 608 if the value of **`LOGONLIM`** is exceeded, or EXCI return code 202 if the number of CICS receive sessions is exhausted.
- Significant numbers of requests are being timed out by the EXCI timeout mechanism.
- Requests receive EXCI return code 202 if the number of CICS receive sessions is exhausted.

The EXCI options table, generated by the DFHXCOPT macro, enables you to specify the time interval, in hundredths of a second, during which the EXCI waits for a DPL command to complete.

If `CS_CALLOC = CS_SLOGONLIM` all EXCI pipes are in use. Resource shortage errors might occur because there are not enough EXCI pipes available to support the number of incoming requests from the worker threads.

If several CICS servers are being used by each Gateway, change the CICS Transaction Gateway pipe usage model by setting the **`CTG_PIPE_REUSE=ONE`** configuration parameter.

If large numbers of worker threads are being used, increase the value of the CICS **`LOGONLIM`** system parameter for the LPAR.

If large numbers of worker threads are in use, or significant numbers of requests are being timed out by the EXCI timeout mechanism, and the value of the **`RECEIVECOUNT`** parameter in the CICS SESSIONS definition is less than double the value of the number of worker threads created (WT_CCURR), increase the value of the **`RECEIVECOUNT`** parameter in the CICS SESSIONS definition to be at least double the value of WT_CCURR.

Other factors

When you define your EXCI connections you can choose between *generic EXCI pipes* and *specific EXCI pipes*. Specific EXCI pipes give slightly better performance. See "CICS server connection definition" on page 122 for more information on selecting your EXCI connections.

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 "Tuning the gateway to avoid out of memory conditions" on page 61 for more information.
- Increase the stack available to the Java threads using the **`-Xss`** parameter. The default value of -Xss is 256 KB with Java V7 31-bit, and 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 Servers.

For information about how statistics might indicate that a JVM is short of heap storage see JVM stress causing poor performance in the Gateway daemon.

While the default maximum heap size (128MB) is adequate for an EXCI workload, it might be necessary to increase this to 256MB or 512MB for an IPIC workload. Ensure that the region size is increased accordingly to match any increases in heap size, otherwise the Gateway daemon might fail with a java.lang.OutOfMemory error. See “Configuring Java shared classes” on page 97.

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 CICS Transaction Gateway 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 z/OS: Programming Guide* for more information.

ECI containers

The number and size of ECI containers have an effect on performance. As you increase the number and size of your containers, you need more system resources to process them, and your response times are longer. Load balancing can help control the flow of your data if you use large containers with multiple simultaneous requests across a single gateway.

Synchronous or asynchronous ECI calls

CICS Transaction Gateway has to do less processing to handle a synchronous ECI call than to handle an equivalent asynchronous call. Also, synchronous ECI calls create fewer network flows than asynchronous calls. This means that synchronous

ECI calls give better performance than asynchronous calls.

Extended logical units of work

Take care when extending a logical unit of work across multiple program link calls that might span a long time period (for example, user thinking time). The logical unit of work holds various locks, and other CICS resources, on the server. This might cause delays to other users who are waiting for the same locks and resources.

Also, each logical unit of work occupies one CICS non-facility task for the duration of its execution. This means that you must define enough free tasks in the CICS server to service the maximum expected number of concurrent calls.

Performance monitoring tools

The performance monitoring tools provide a way of measuring system performance characteristics such as transaction throughput and processor usage.

Resource Management Facility (RMF™)

This tool collects information about activity for selected areas of the system. Information is collected in the form of *System Management Facility (SMF)* records. You can use the tool to measure processor and I/O activity in the CICS Transaction Gateway address space, and also, for example, to measure similar activity in the CICS server and WebSphere. For more information see the *z/OS Resource Management Facility (RMF) Performance Management Guide*, SC33-7992.

Request monitoring exits

These exits are applications written by independent software vendors that can be called at significant points in the request flow through the Gateway daemon and Gateway classes. Request monitoring exits are an important tool used when analyzing transaction flows during problem determination and performance tuning.

IBM Tivoli OMEGAMON XE for CICS

This tool monitors and manages CICS transactions and resources. IBM Tivoli OMEGAMON XE for CICS quickly detects and isolates problems when they occur on CICS systems.

Refer also to the performance and tuning documentation for WebSphere, TCP/IP, CICS Transaction Server for z/OS, and TXSeries and the documentation supplied with your operating system.

Statistics and performance assessment

Use statistics to assess system performance and to identify and resolve performance problems.

Indicators of performance problems are poor response time, and out of memory conditions. Statistics provide the information needed to improve system performance.

Investigating poor response times

Different system configurations and loads can lead to poor response times. Use the statistics provided by CICS Transaction Gateway to identify possible causes of poor response times and improve them.

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. You might need to increase the EXCI pipe limits, see EXCI resource shortage. 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.

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 and the associated region size

Note: SE_IGCTIME is measured in milliseconds and GD_IRUNTIME is measured in seconds.

Tracing

Full tracing of CICS Transaction Gateway can degrade system performance and should not be used in a production environment.

Where possible, try to measure response times through the different parts of your system, without using tracing, to find where delays are happening. For example, you can measure response times at the client application, and also through CICS and WebSphere. For more information, see “Investigating poor response times” on page 70.

Chapter 7. High availability

High availability is a key feature of CICS Transaction Gateway on z/OS. With a system solution that delivers high availability, any single failure does not cause failure of the total solution and increased capacity can be provided by adding additional components. High availability is available for all types of ECI request including SYNCONRETURN, extended LUW, two-phase commit XA transactions and all types of ESI request.

High availability of the Gateway daemon uses TCP/IP load balancing to spread connections across a highly available Gateway group. You can also increase the efficiency of your load balancing function by taking account of CICS server status provided by dynamic feedback to the IBM z/OS Workload Manager (WLM).

Related concepts:

"Health monitoring" on page 74
A TCP/IP load balancer that is allocating a connection to the Gateway daemon detects whether or not a CICS server is available. The Gateway daemon reports the health of its CICS server connections to the TCP/IP load balancer.

"TCP/IP load balancing"
Load balancing ensures high system availability through the distribution of workload across multiple components.

"Dynamic server selection" on page 77
Dynamic server selection (DSS) provides the CICS Transaction Gateway administrator with several mechanisms for dynamically controlling the flow of work to CICS servers in a high availability operating environment. DSS also provides flexibility for deployment or change in environment.

Related tasks:

"Determining health status" on page 260
The current health status is available in the GD_CHEALTH statistic. This information describes how to find the current health status.

TCP/IP load balancing

Load balancing ensures high system availability through the distribution of workload across multiple components.

TCP/IP load balancing provides high availability of the Gateway daemon by distributing connections from clients across multiple Gateway daemons. The efficiency of load balancing can also be increased by taking into account CICS server status provided by dynamic feedback to the IBM z/OS Workload Manager (WLM).

When TCP/IP load balancing is being used, a connection request from a Client application is received through a network socket by the TCP/IP load balancing component. This component decides which Gateway daemon in the Gateway group receives the request. A connection is established to the chosen Gateway and, when it is established, requests from the Client application continue to go through the same connection to the chosen Gateway daemon. The Client application cannot determine which Gateway daemon instance is selected.

If a problem occurs, such as a temporary network failure, or an unplanned shutdown of CICS Transaction Gateway or WebSphere Application Server, the existing connection is lost. The next time a request is received, a new connection is created. This connection can go to any Gateway daemon instance in the Gateway group.

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

To support TCP/IP load balancing for XA requests a highly available Gateway group (HA group) must be defined. For more information, see "Highly available Gateway group" on page 76.

The use of the APPLID qualifier has implications for CICS connections. For more information, see IPIC server connections.

Port sharing

TCP/IP port sharing enables requests for work to be shared between several Gateway daemons through a single TCP/IP port. Port sharing provides TCP/IP load balancing in a single LPAR.

When connections are established between a Client application and a Gateway daemon, the TCP/IP port sharing component of the z/OS TCP/IP Communications subsystem distributes requests across multiple Gateway daemons. When the connection is established, subsequent requests from the Client application continue to use the same connection.

TCP/IP port sharing is included in Scenario SC03.

Sysplex Distributor

Sysplex Distributor is the strategic IBM solution for connection workload balancing across a z/OS sysplex. Sysplex Distributor provides TCP/IP load balancing across multiple LPARs.

When a connection between a Client application and a Gateway daemon is established, Sysplex Distributor distributes requests across multiple Gateway daemons. When the connection is established, requests from the Client application continue to use the same connection.

Sysplex Distributor can be combined with TCP/IP port sharing.

For more information about Sysplex Distributor, see the publications in the z/OS Communications Server library at:

http://www.ibm.com/software/network/commserver/zos/library/.

Health monitoring

A TCP/IP load balancer that is allocating a connection to the Gateway daemon detects whether or not a CICS server is available. The Gateway daemon reports the health of its CICS server connections to the TCP/IP load balancer.

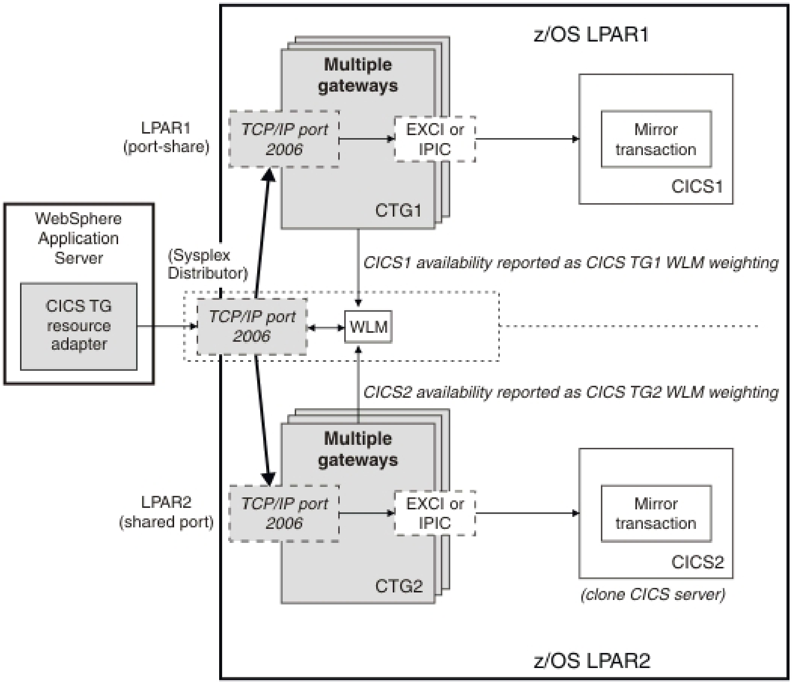
The best results are obtained if one Gateway daemon connects to one CICS server. If a Gateway daemon connects to more than one CICS server, a failure in one CICS server may prevent work from being sent to the others.

Sysplex Distributor and TCP/IP port sharing can both use health monitoring when allocating new connections. If you do not activate health reporting in the Gateway daemon, statistics are still collected by the Gateway daemon, but are not reported to the TCP/IP load balancer.

Health reporting is effective exclusively in TCP/IP load balancing topologies with CICS Transaction Gateway running in remote mode. Over intervals specified by the health interval setting, the Gateway daemon monitors certain error codes to determine the health of communications with CICS. The TCP/IP load balancer then prioritizes the creation of new incoming client application connections to Gateway daemons in the load balancing group. Gateway daemons reporting a higher health value receive a greater proportion of the incoming connections than those reporting a lower health value.

Health monitoring in a Sysplex with Sysplex Distributor

The diagram shows Gateway daemons reporting on the availability of CICS servers to IBM Workload Manager.



How health is calculated

The Gateway daemon health interval defines the amount of time, in seconds, that the Gateway daemon monitors particular error codes to determine the health of communications with CICS. The default health interval is 60 seconds. If no connectivity problems occur, the Gateway daemon health remains at 100.

Intermittent problems can cause the health of communications with CICS to drop which, in turn, causes the load balancer to reduce the amount of work sent to the CICS server affected. If the problem disappears, health recovers.

If the health of communications with CICS drops to zero, the Gateway daemon issues a warning message, and the load balancer stops sending connection requests to the Gateway daemon until the health value has been reset by a Gateway daemon administrator.

These CICS return codes indicate that a request failed because of problems with the health of communications with CICS:

- ECI_ERR_NO_CICS
- ECI_ERR_RESOURCE_SHORTAGE
- ECI_ERR_SYSTEM_ERROR
- ESI_ERR_NO_CICS
- ESI_ERR_RESOURCE_SHORTAGE
- ESI_ERR_SYSTEM_ERROR

The health of communications with CICS represents the percentage of requests during the health interval that succeeded. If all requests succeed, health of communications with CICS is 100. If 30% of requests fail, health is 70. If there are fewer than 20 requests in the interval, each failing request reduces health by 5, however the health of communications with CICS can never drop below zero.

The table shows how health can fluctuate:

Table 4. Health fluctuation

Event	Requests processed	Requests failed	System health
Health interval 1	1000	200	80%
Health interval 2	0	0	80%
Health interval 3	500	50	90%
Health interval 4	15	1	95%
Health interval 5	200	0	100%

Related concepts:

"Health reporting" on page 155
The Gateway daemon can monitor certain error codes to determine the health of communications with CICS.

Related tasks:

"Determining health status" on page 260
The current health status is available in the GD_CHEALTH statistic. This information describes how to find the current health status.

Related information:

"Resetting health status" on page 260
This information describes how to reset the health status to 100.

Highly available Gateway group

A highly available (HA) group is a Gateway group that uses TCP/IP load balancing. You can regard an HA group as a single logical Gateway daemon.

Gateway daemons are defined as belonging to a highly available (HA) group by definition of a common APPLID qualifier. The use of an APPLID qualifier must also be taken into account, when configuring IPIC server connections.

Each Gateway daemon in an HA group must have identical configuration details for:

- TCP and SSL protocol handlers
- IPIC server definitions
- Logical CICS server names
- CICS request exit configuration
- Security settings
- XA support

Gateway daemon instances can resolve XA transactions on behalf of other Gateway daemon instances in the same HA group. For more information, see Sysplex restrictions.

For more information about configuring CICS Transaction Gateway in an HA group to support XA transactions see Configuring for XA transaction support .

Dynamic server selection

Dynamic server selection (DSS) provides the CICS Transaction Gateway administrator with several mechanisms for dynamically controlling the flow of work to CICS servers in a high availability operating environment. DSS also provides flexibility for deployment or change in environment.

DSS supports one-phase commit transactions, two-phase commit transactions, and ESI requests.

DSS is provided by a default CICS server, policy-based DSS definitions, logical CICS server definitions (deprecated), or a CICS request exit. You can use a default CICS server either on its own, or in combination with one of the other DSS mechanisms. Policy-based DSS, logical CICS server definitions (deprecated) and the CICS request exit cannot be used in combination with each other.

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 148.

Policy-based dynamic server selection

Policy-based dynamic server selection (DSS) provides a flexible mechanism for controlling the flow of work to CICS servers.

In a high availability topology that uses policy-based DSS, workload can be dynamically managed across a number of CICS servers by using round-robin workload distribution; alternatively one or more backup servers can be made available to a CICS server, if that server becomes unavailable (failover).

Policy-based DSS enables failed requests to be automatically retried on alternative CICS servers during abnormal conditions, for example during planned maintenance or overnight batch processing, and for restarted CICS servers to become available for work again, without operator intervention.

At runtime, policy-based DSS maps logical CICS server names to CICS server groups, where a group definition includes a preassigned workload distribution algorithm. A request specifying a logical CICS server name that arrives at CICS Transaction Gateway is routed to one of the servers in the CICS server group that maps to the logical CICS server name, in accordance with the workload distribution algorithm.

Blank server names can be mapped explicitly to a DSS server group or can be configured to use a single default CICS server.

Server names for which no explicit mapping has been defined can be mapped to a default DSS server group; this mapping includes blank server names unless there is an explicit mapping for them. Such a mapping overrides any default server definition.

For information about how to configure a DSS policy see “Configuring a dynamic server selection policy” on page 148.

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 the CICS request exit does not specify a CICS server name and a default CICS server has not been defined, CICS Transaction Gateway assumes that the EXCI protocol is being used, and delegates the decision about which CICS server to use, to EXCI. The EXCI interface can then optionally use DFHXCURM to select a target CICS server.

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

CICS request exit and DFHXCURM

If you are using dynamic server selection, the CICS request exit provides a number of advantages when compared with the DFHXCURM user replaceable module (URM).

The CICS request exit is part of the Gateway daemon and can be used for redirecting requests over EXCI or IPIC connections. Because the CICS request exit supports IPIC, you can use it to redirect ECI requests calling CICS channel based programs and ESI requests which are not supported over EXCI connections.

The DFHXCURM user replaceable module is used for redirecting requests over EXCI in a local mode topology. DFHXCURM cannot be used with IPIC connections.

Benefits of using the CICS request exit

When implementing dynamic server selection, the CICS request exit provides these advantages over DFHXCURM:

- The CICS request exit is integrated with the statistics and request monitoring exits.
- The CICS request exit is invoked for every transaction rather than at pipe allocation, and provides the ability to quickly discover when failed CICS servers are restarted.
- CICS Transaction Gateway can track the number of times a CICS request exit is called; If the value is greater than the number of transactions that have been processed, this provides an indication that some requests were retried.
- CICS Transaction Gateway can log statistics about the work sent to each CICS server that the CICS request exit selected.
- If a request fails with an error at the start of a transaction, the CICS request exit can be called a second time with details of the previous failure.

Chapter 8. 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.

For information about the remote and local mode topologies, see Deployment topologies.

Configuring the system environment

To configure the system environment, you set the system processing and file system parameters in the BPXPRMxx parmlib member.

Increasing the MAXCPUTIME value

MAXCPUTIME is the time limit (in seconds) for processes created by rlogind, telnetd, and other daemons.

When running from batch, the RACF user ID that starts the job is associated with a USS ID in the RACF OMVS segment. If you set CPUTIMEMAX in the OMVS segment for the RACF user ID, the Gateway daemon running under that ID takes the value specified for CPUTIMEMAX in the OMVS segment. If CPUTIMEMAX is not set in the OMVS segment, the system wide value for MAXCPUTIME that has been set in BPXPRMxx is used.

You can set a system-wide limit in BPXPRMxx and then set higher limits for individual users. Use the RACF ADDUSER or ALTUSER command to specify the CPUTIMEMAX limit for individual processes.

If the volume of traffic is high, it might be necessary to change the MAXCPUTIME value in the BPXPRMxx member in SYS1.PARMLIB.

You can set this value on a user ID basis using the CPUTIMEMAX value in the OMVS segment of the RACF user ID profile. For more information, see *z/OS UNIX System Services Planning*.

Configuring a local mode topology

The configuration tasks required for a local mode topology.

About the local mode topology

The local mode topology on z/OS is typically used with WebSphere Application Server, using the ECI resource adapter to connect directly to a CICS server. However, it is also possible to use a Java client application from UNIX System Services. Both situations require the same configuration settings but differ in where those configuration settings are specified. Configuration settings also vary depending on whether or not the EXCI protocol might be used to connect to CICS. If the EXCI protocol is not required, the only configuration required is for the native library path.

For more information about how to define the local mode configuration settings in WebSphere Application Server for z/OS see “Deploying the ECI resource adapter on WebSphere Application Server for z/OS” on page 173.

The sample JCL job SCTGSAMP(CTGTESTL) provides an example of configuring local mode for EXCI.

Local mode configuration settings

Configure the library path by setting the **`LIBPATH`** environment variable to the product `/bin` directory. For example, the following entry might be set in the STDENV section of the CTGTESTL job:

```
LIBPATH=/usr/lpp/cicstg/ctg900/bin
```

When using the EXCI protocol to connect with CICS, the following additional environment variables must be set:

Environment variables to configure local mode to use the EXCI protocol:

- CTG_EXCI_INIT determines whether or not EXCI is loaded (default is YES if undefined).
- CTG_PIPE_REUSE determines how allocated EXCI pipes are reused.
- DFHJVPIPE defines the name of the pipe that CICS Transaction Gateway uses for EXCI calls.
- DFHJVSYSTEM_nn defines the name and description of an EXCI connected CICS server to be returned in response to a request for the CICS_EciListSystems function.
- STEPLIB identifies the library containing the default EXCI options and the EXCI load modules.

Path to include Java command:

When using a local mode Java client from UNIX System Services, the PATH might require updating to include the **`java`** command. For example, the following entry might be set in the STDENV section of the CTGTESTL job:

```
PATH=<java_path>/bin
```

For more information about these environment variables see “STDENV file” on page 98.

Configuring a remote mode topology

In a remote mode topology, you configure the Gateway daemon on the system on which CICS Transaction Gateway is installed and deploy and configure remote client applications.

Before starting CICS Transaction Gateway the values in the configuration files must be correctly defined. After editing the configuration files, you must restart the Gateway daemon for the changes to the configuration file to take effect.

Configuring the Gateway daemon

The Gateway daemon is configured by creating a configuration file and a set of environment variable definitions, defined by the STDENV DD card of the Gateway daemon JCL CTGBATCH job step. See “STDENV file” on page 98 for more information. The configuration file can reside on the USS file system or as an MVS dataset on the MVS file system, with the path or MVS dataset defined by the environment variable CICSCLI. If the environment variable CICSCLI is not defined, the Gateway daemon will look for a configuration file, ctg.ini, in the product \bin directory. This is the historical default location, however, it is recommended to use another location outside the SMP/E-managed product install path.

You can specify a different location and optionally change the name of the configuration file ctg.ini by setting the CICSCLI environment variable to point to a USS file:

```
CICSCLI=/u/userid/myconfig.ini
```

Figure 9. Setting the CICSCLI environment variable to point to a USS file

If the configuration is maintained in MVS and edited using ISPF, you can set the CICSCLI environment variable to point to the MVS dataset name or dataset location:

```
CICSCLI=//'HLQ.QUAL.PDS(MEMBER)'
```

Figure 10. Setting the CICSCLI environment variable to point to an MVS dataset

Use the double forward slash symbol // to denote that the configuration is held in an MVS dataset not a file. The CICS Transaction Gateway startup message CTG8400I indicates the location of the configuration if it is in a dataset that uses the same notation. A sample configuration on MVS is provided in dataset member CTGCONF, in the CTGSAMP dataset which is installed as part of the product. If a value is not specified for this environment variable, the default path <install_path>/bin/ctg.ini is used during startup. The maximum field length is 100 characters.

At startup, the name and location of the configuration file being used is written to the Gateway information log.

Related information:

"Configuration parameter reference" on page 156
The way in which you configure CICS Transaction Gateway depends on how the Gateway daemon is to be started.

"Environment variables: local and remote mode" on page 100
Environment variables available for use with local mode and remote mode topologies.

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

Edit the GATEWAY section of the configuration file to configure the Gateway daemon resources.

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 **ctgstart** command with the **-initconnect** option to override the value of the **port** parameter. For more information, see "Options on the ctgstart command" on page 254

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

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

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 60.

You can use the **ctgstart** command with the **-maxconnect** option to override the value of the **port** parameter. For more information, see "Options on the ctgstart command" on page 254

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

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

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 **ctgstart** command with the **-initworker** option to override the value of the **port** parameter. For more information, see "Options on the ctgstart command" on page 254

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

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

Maximum number of worker threads:

The **maxworker** parameter defines the maximum number of concurrent 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. Set the value to be less than or equal to `maxconnect`. If the value exceeds the EXCI pipes limit, a warning message is issued, and the value of **maxworker** is reduced to be equal to the EXCI logon limit, **LOGONLIM**. Because all CICS server connections share the same pool of worker threads, this restriction also affects IPIC connections. If EXCI is disabled by setting environment variable CTG_EXCI_INIT to NO, this restriction does not apply.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 60.

You can use the **ctgstart** command with the **-maxworker** option to override the value of the **port** parameter. For more information, see "Options on the ctgstart command" on page 254

Because there is a maximum limit on the number of EXCI pipes per MVS address space, set **maxworker** according to the EXCI pipe reuse model:

Reuse all (CTG_PIPE_REUSE=ALL)
Less than or equal to the EXCI pipe maximum limit set by CICS (LOGONLIM) divided by the number of unique APPLIDs in communication with CICS Transaction Gateway.

Reuse one per thread (CTG_PIPE_REUSE=ONE)
Less than or equal to the EXCI pipe maximum limit set by CICS (LOGONLIM).

In both cases, to avoid resource shortage errors set RECEIVECOUNT in the CICS sessions definition to be greater than the number of EXCI pipes being used. For more information see "EXCI pipe limit exceeded" on page 276. Starting multiple instances of CICS Transaction Gateway can increase throughput, for more information see "Starting multiple CICS Transaction Gateways" on page 254.

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

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

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.

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

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.

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

Enable reading input from SDSF:

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

`noinput=<on|off>`

Description
: Set the value to **on** to disable reading of the input from the MVS console. If you intend to use the CTGBATCH utility to run the CICS Transaction Gateway set the **`noinput`** parameter to **on**. If you intend to run **`ctgstart`** command under USS set the **`noinput`** parameter to **off**.

You can use the **`ctgstart -noinput`** command to override the value of **`noinput`**. For more information, see "Options on the ctgstart command" on page 254.

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

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

Gateway daemon logging

Edit the GATEWAY section of the configuration file to configure the Gateway daemon logging resources.

Information log messages are written to standard out, Warning and Error log messages are written to standard error. These destinations can be configured further when the Gateway daemon is started using CTGBatch. See "Writing messages to the JES logs" on page 253 for more information.

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.

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

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.

Default value
The default for this parameter is off.

Display TCP/IP hostnames:

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 **`ctgstart`** command 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.

Default value
The default for this parameter is off.

TCP protocol settings

The parameters that you use to define a TCP protocol handler for remote client connections.

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 159 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.

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.

You can use the **`ctgstart`** command with the *-port* option to override the value of the **`port`** parameter. For more information, see "Options on the ctgstart command" on page 254.

This parameter is in the "TCP protocol parameters" on page 159 subsection of the GATEWAY section of the configuration file.

Default value
If this parameter is not specified, the default value is 2006.

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 159 subsection of the GATEWAY section of the configuration file.

Default value
: If this parameter is not specified, the default value is 2000 milliseconds.

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 90 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 159 subsection of the GATEWAY section of the configuration file.

Default value
: If this parameter is not specified, the default value is 600000 milliseconds.

Ping frequency interval:

The **`pingfrequency`** parameter specifies the frequency that ping messages are sent by the Gateway daemon to attached Java and .NET 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 90 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 159 subsection of the GATEWAY section of the configuration file.

Ping messages are not sent to C Client applications.

Default value
If this parameter is not specified, the default value is 60000 milliseconds.

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 159 subsection of the GATEWAY section of the configuration file.

Default value
By default, this parameter is not included in the protocol handler parameters.

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 159 subsection of the GATEWAY section of the configuration file.

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

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 159 subsection of the GATEWAY section of the configuration file.

For more information, see the *CICS Transaction Gateway: Application Programming Guide*.

Default value
By default, this parameter is not included in the protocol handler parameters.

SSL protocol settings

The parameters that you use to define an SSL protocol handler for remote client connections.

Note: For a description of the parameters **`Use hardware cryptography`**, **`key ring location`**, **`key ring file`**, **`key ring password`**, and **`key ring password encryption`**, see SSL key ring configuration.

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 160 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.

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.

You can use the **`ctgstart`** command with the *-sslport* option to override the value of the **`port`** parameter. For more information, see “Options on the ctgstart command” on page 254.

This parameter is in the SSL protocol parameters subsection of the GATEWAY section of the configuration file.

Default value
If this parameter is not specified, the default value for SSL is 8050.

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 160GATEWAY section of the configuration file.

Default value
If this parameter is not specified, the default value is 2000 milliseconds.

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 90 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 160GATEWAY section of the configuration file.

Default value
If this parameter is not specified, the default value is 600000 milliseconds.

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 90 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 160 subsection of the GATEWAY section of the configuration file.

Default value
If this parameter is not specified, the default value is 60000 milliseconds.

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 160 subsection of the GATEWAY section of the configuration file.

Default value
By default, this parameter is not included in the protocol handler parameters.

Socket close delay:

The **solinger** parameter sets the delay value in seconds for closing a 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 "SSL protocol parameters" on page 160 subsection of the GATEWAY section of the configuration file.

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

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 160 subsection of the GATEWAY section of the configuration file.

For more information, see the *CICS Transaction Gateway: Application Programming Guide*.

Default value
By default, this parameter is not included in the protocol handler parameters.

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 160GATEWAY section of the configuration file.

Default value
By default, this parameter is not included in the configuration file.

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 160 subsection of the GATEWAY section of the configuration file.

Default value

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

Configuring trace settings

Edit the trace attributes in the GATEWAY section of the configuration file.

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 name without a path, the file is created in the <install_path>/bin 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 **`ctgstart -tfile=`***pathname* command to override the value of **`tfile`**. For more information, see Command reference.

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

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 **`ctgstart -tfilesize=`***number* command to override the value of **`tfilesize`**. For more information, see Command reference.

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

Default value

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

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 **`ctgstart -dumpoffset=`***number* command to override the value of **`dumpoffset`**. For more information, see "Options on the ctgstart command" on page 254.

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

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

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 **`ctgstart -truncationsize=`***number* command to override the value of **`truncationsize`**. For more information, see "Options on the ctgstart command" on page 254.

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

Default value
If this parameter is not specified, the default value is 80.

Enable Gateway daemon trace on startup:

Set the **`trace`** parameter to **on** to enable tracing of the Gateway daemon.

This is equivalent to specifying the **-trace** option on the ctgstart command.

The default setting is **off**.

This setting is case insensitive.

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

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 **`ctgstart -stack`** command to override the value of **`stack`**. For more information, see "Options on the ctgstart command" on page 254.

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

Default value
The default for this parameter is off.

JNI trace:

The *CTG_JNI_TRACE* and *CTG_JNI_TRACE_ON* environment variables are used to control JNI trace.

See “Environment variables: local and remote mode” on page 100 for more information.

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.

JVM options can be specified in the parameters passed to the **`ctgstart`** command using the **`PARM`** string on the `EXEC PGM=CTGBATCH` section of the JCL, or using the *CTGSTART_OPTS* environment variable. Each JVM option must be prefixed with `-j`.

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

```
CTGSTART_OPTS=-j-Xmx512M
```

The parameter list for the *CTGSTART_OPTS* environment variable can exceed 100 characters. For more information see “Environment variables: remote mode” on page 102.

Configuring Java shared classes

CICS Transaction Gateway uses the Java shared classes feature to improve performance.

Java shared classes provides a transparent and dynamic way of sharing loaded classes that places no restrictions on Java Virtual Machines (JVMs) that are sharing class data. Java shared classes helps reduce virtual memory usage and can improve startup time. CICS Transaction Gateway creates a class cache called cicstg*vrm%g*, where *vrm* is the version, release and modification level of CICS Transaction Gateway, and *%g* is the group name of the user that is running the Gateway daemon job. The cache is created in the /tmp/javasharedresources directory unless the location is overridden with the cachedir sub-option. The class cache options used by the Gateway daemon are output in message CTG6134I during initialization.

The available sub-options are:

groupAccess
Sets the operating system permissions on a new cache to allow group access to the cache. The default is user access only.

nonpersistent
Creates a nonpersistent cache in shared memory. The shared memory (and the cache) are lost when the operating system shuts down. Nonpersistent and persistent caches can have the same name. You must always use the nonpersistent suboption when running utilities such as destroy on a nonpersistent cache. The default is persistent.

nonfatal
Ensures that the JVM ignores any potentially fatal permission problems or disk space problems, and attempts to start normally and to operate

normally. For example a JVM does not attempt to open a readonly cache unless a JVM read/write operation on that cache previously failed.

By default, the following class cache options are specified when the Gateway daemon starts:

```
-Xshareclasses:name=cicstgvrm%g,groupAccess,nonpersistent,nonfatal
```

For more information on the Java shared class cache options see your Java documentation.

To override the options used by CICS Transaction Gateway for the -Xshareclasses, set the CTGSTART_OPTS CICS TG environment variable, for example:

```
CTGSTART_OPTS=-j-Xshareclasses:name=ctgtest%g,cachedir=/tmp/ctgtest,groupAccess,
nonpersistent,nonfatal
```

To disable class caching for the Gateway daemon, specify the following option:

```
CTGSTART_OPTS=-j-Xshareclasses:none
```

To list all the valid sub-options to -Xshareclasses, type the following in the bin directory of your Java installation:

```
java -Xshareclasses:help
```

If Gateway daemons using the same Java major version are started on the same LPAR by different user IDs, you must ensure that all user IDs have access to the relevant class cache entities. For more information see your Java documentation.

Setting environment variables

Environment variables control how the CICS Transaction Gateway functions in both local and remote modes.

To set environment variables for CICS Transaction Gateway for z/OS use the STDENV file, the ctgenvvar script, or explicitly export the environment variables.

STDENV file

The STDENV file can be used if the Gateway daemon is to be started in batch mode using CTGBATCH to define the required environment variables.

When using CTGBATCH to start the Gateway daemon, if a ctgenvvar file exists in the default install directory, the settings in the ctgenvvar file are used in preference to those defined in the STDENV file. This is because the ctgenvvar file is referenced within the ctgstart script.

Use the conversion script ctgconvenv, to convert environment variables, previously set by the ctgenvvar script, into a STDENV file that can then be used by the CTGBATCH program.

The STDENV file can be an MVS sequential file, a PDS member, or an HFS file and has the following syntax rules:

- Any line with a hash (#) in column one is interpreted as a comment.
- Blank lines are ignored.
- Leading spaces are removed from a line
- An entry for a name/value pair can start in any column.

- A valid entry for a name/value pair must not have spaces between the name, equals sign or value.
- Use backslash (\) as a continuation character.
- Trailing spaces after a continuation character are removed.
- A continuation character can be placed in any column.
- A continuation character must be the last non-whitespace character on a line.
- A value containing continuation characters is deemed to be completed by the first line that omits the continuation character.
- Trailing spaces are removed from single-line data and from the end of concatenated multi-line data
- Do not enclose value strings in single or double quotes. Every character, after the first equals sign, that is not a trailing space or continuation character is assumed to be part of the value data.

A sample STDENV file is in the SCTGSAMP library as a member CTGENV.

Related information:

"Starting in batch mode" on page 249
The recommended way of running the CICS Transaction Gateway for z/OS Gateway daemon as a production system is in batch mode. CTGBATCH is a utility, used to launch USS programs through the MVS batch environment and route stdout and stderr I/O to MVS destinations.

The ctgenvvar script

Use the ctgenvvar script if the Gateway daemon is to be started via the USS command line for example, in a test environment.

The ctgenvvar script can be executed alone or referenced by the ctgstart script to set the variables. You can create the ctgenvvar script by editing the supplied ctgenvvarsamp script, and then copying or renaming to ctgenvvar, see "Configuration parameter reference" on page 156.

Because the files under <install_path>/bin are managed by SMP/E, this area is regarded as read-only. Therefore, the customized ctgenvvar script is likely to be located under a user-writable HFS path.

When running the Gateway daemon via the ctgstart script the environment variable "CTGENVVAR", if set, will be used to define the fully qualified location of a ctgenvvar script (including the file name of the script). If it is not set, ctgstart will check for the script "ctgenvvar" under HFS path <install_path>/bin (for compatibility with previous versions). For example,

```
export CTGENVVAR=/u/ctgusr/ctg700cfg/ctgenvvar
```

If neither of these locations contains a script, ctgstart will assume that the environment variables are already in place, and attempt to start the Gateway daemon.

The target script itself must have the '+x' executable file attribute set at the appropriate level for the current user.

After you have installed the product, you can use `ctgenvvar` to establish the environment that Java client applications

running in local mode will use, for example:

```
. <config_path>/ctgenvvar
```

where <config_path> is the HFS location of the customized configuration script.

Because by default `ctgenvvar` searches for `ctgstart`, ensure that `<install_path>/bin` is in the path for the current environment. You can edit ctgenvvar to define where ctgstart is installed.

Environment variables: local and remote mode

Environment variables available for use with local mode and remote mode topologies.

AUTH_USERID_PASSWORD
Specifies whether the user ID and password is authenticated with the External Security Manager (ESM) such as RACF. This setting applies exclusively to EXCI connections to CICS. Set this variable to YES to enable authentication by the ESM. Set it to NO if authentication by the ESM is not required. If the variable is set to any value other than NO, authentication is enabled. If the variable is not set, authentication is not enabled.

If you set this variable see also “Configuring for client certificate mapping” on page 143.

COLUMNS

Specifies the maximum line-length for messages output to the console when CICS Transaction Gateway is started. If this variable is not set, the line-length is set to 40 columns. The minimum line-length is 40 columns. The maximum possible line-length is 160 columns.

CTG_EXCI_INIT
Specifies whether or not EXCI is loaded. If this variable is set to YES, EXCI is loaded. If the variable is set to NO, EXCI is not loaded. The default value is YES.

If EXCI is not active:

- An information message is logged.
- Attempts to make EXCI calls result in an ECI_ERR_NO_CICS return code.
- Attempts to use an EXCI server result in a communications failure count being incremented.
- ListSystems calls do not include EXCI servers.

CTG_JNI_TRACE
Sets the name of the JNI trace file. This environment variable only defines the name of the JNI trace file; it does not enable trace. The default file name is ctgjni.trc. 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
Specifies whether or not JNI trace is enabled. Set this environment variable to YES to enable JNI trace when the CICS Transaction Gateway is started. Tracing is not enabled if the variable is set to any value other than YES.

CTG_MIXEDCASE_PW
Specifies whether or not mixed-case passwords are authenticated. Set this environment variable to YES to enable mixed-case passwords to be authenticated by an external security manager such as RACF. If you enable support for mixed-case passwords, you must also set environment variable

AUTH_USERID_PASSWORD to YES. CTG_MIXEDCASE_PW must be set to YES to allow password phrases to be authenticated.

If CTG_MIXEDCASE_PW is set to NO, CICS Transaction Gateway converts all passwords to upper case before authenticating with the ESM.

Valid values are YES|NO. Any other value is treated as not valid, and the default value used instead.

CTG_PIPE_REUSE

Specifies whether all allocated EXCI pipes are reused by CICS Transaction Gateway, or only a maximum of one is reused per worker thread.

ALL

Reuse all allocated pipes. This is the default.

Specify this option if maximum throughput is the priority.

If the pipe that is allocated to a given worker thread is not for the correct server for the next ECI request, using this pipe model, the pipe remains allocated and a new pipe is allocated for the new server connection on that worker thread. This model limits the maximum number of worker threads to the EXCI pipe limit divided by the maximum number of servers.

ONE

Reuse a maximum of one allocated EXCI pipe for each worker thread.

Specify this option if the maximum number of concurrent worker threads is the priority.

If the pipe that is allocated to a given worker thread is not for the correct server for the next ECI request, using this pipe model, the pipe is deallocated and then reallocated to the new server. This model limits the maximum number of worker threads to the EXCI pipe limit.

Abbreviations are not allowed. An unrecognized value causes the default of ALL to be used.

CTG_SWAPPABLE

Specifies whether the address space where CICS Transaction Gateway runs is swappable or nonswappable. Do not set this environment variable unless CICS Transaction Gateway runs in local mode. If you do not set the environment variable, CICS Transaction Gateway runs in a nonswappable address space. If you set the environment variable to YES, CICS Transaction Gateway runs in a swappable address space.

If a Java Client application makes an ECI request while CICS Transaction Gateway is swapped out, the request fails with these error messages:

```
CCL6668E: Initial handshake flow failed
CCL6666E: Unable to flow request to the Gateway
```

To avoid this, CICS Transaction Gateway is made nonswappable after the first ECI request has been successfully flowed. CICS Transaction Gateway can still be swapped out before the first ECI request is flowed. If this happens, ECI requests fail until z/OS swaps CICS Transaction Gateway back in.

Nonswappable mode

You are recommended to run CICS Transaction Gateway in a nonswappable address space, unless it is running in local mode. (It always runs in swappable mode under WebSphere Application Server.)

To make CICS Transaction Gateway nonswappable, ensure that the user running the CICS Transaction Gateway process has READ access to the BPX.STOR.SWAP FACILITY class. No other action is required. If CICS Transaction Gateway cannot be made nonswappable, a log message is output, and CICS Transaction Gateway runs in a swappable address space.

Swappable mode

You are advised to run CICS Transaction Gateway in a swappable address space if it will run in local mode. To do this, set the CTG_SWAPPABLE environment variable to YES. If you set this variable to any other value CICS Transaction Gateway runs in a nonswappable address space.

When an application makes an address space nonswappable, it might cause additional real storage in the system to be converted to preferred storage. Because preferred storage cannot be configured offline, using this service can reduce the ability of the installation to re-configure storage in the future. For more information see *z/OS UNIX System Services Planning*, GA22-7800.

DFHJVPIPE

Specifies the name of the specific pipe that CICS Transaction Gateway uses for EXCI calls. If this variable is not set, or is set to a blank value by a command such as `export DFHJVPIPE=""`, CICS Transaction Gateway uses a generic pipe.

DFHJVSYSTEM_*nn*

Specifies the name and description of an EXCI connected CICS server to be returned in response to a request for the **list systems** function. You can set up to 100 environment variables of the form DFHJVSYSTEM_*nn*, where *nn* ranges from 00 to 99. They are listed in order from 00 to 99 with missing numbers ignored. This function call takes a maximum number of systems returned value; if the number of systems defined exceeds that value the call returns the error ECI_ERR_MORE_SYSTEMS.

The value is a string containing the APPLID of the CICS system, followed by a hyphen (-), followed by a description of the CICS system. The description must not be more than 60 bytes long. For example if you specify:

```
DFHJVSYSTEM_00=MYCICS-Test CICS system
```

List systems returns details of a CICS system called `MYCICS` with description `Test CICS system`.

STEPLIB

Specifies the library containing the default EXCI options and the EXCI load modules. Ensure that the STEPLIB environment variable includes any load library that contains EXCI options table DFHXCOPT ahead of the CICS EXCI load library SDFHEXCI in the concatenation order, because the default table supplied is contained in SDFHEXCI.

The maximum length of this field is 200 characters.

Environment variables: remote mode

Environment variables available for use with remote mode topologies.

_BPX_SHAREAS

Specifies whether or not all processes involved in starting the Gateway daemon run in a single address space. Set this variable to YES to specify that all processes involved in starting the Gateway daemon run in a single address space. Set this variable to NO to allow multiple address spaces.

The default value is YES.

_BPXK_SETIBMOPT_TRANSPORT

Specifies the job name of the TCP/IP stack to be used by the Gateway daemon. This environment variable might be useful in installations involving multiple TCP/IP stacks where it is necessary to restrict the Gateway daemon from binding with all TCP/IP stacks. If the specified job name is not active during Gateway daemon initialization and the protocol handler definitions do not specify a bind address, the Gateway daemon protocol handlers are bound to all available TCP/IP stacks.

CICSCLI

Specifies the location of the CICS Transaction Gateway configuration to be used at startup. The value must be one of the following:

- A path and file name, if the configuration is maintained and edited as a configuration file, for example:

  `/u/userid/myconfig.ini`
- An MVS dataset name or dataset location, if the configuration is maintained in MVS and edited using ISPF, for example:

  `CICSCLI=//'HLQ.QUAL.PDS(MEMBER)'`

  Use the double forward slash symbol // to denote that the configuration is held in an MVS dataset not a file. A sample configuration on MVS is provided in dataset member CTGCONF, in the CTGSAMP dataset which is installed as part of the product.

If a value is not specified for this environment variable, the default path <install_path>/bin/ctg.ini is used during startup.

The maximum field length is 100 characters.

The CICS Transaction Gateway startup messages indicate the location of the configuration file. CICS Transaction Gateway does not start if a configuration cannot be found.

CTG_WIDTH

Specifies the maximum width of the STDENV output from the ctgconvenv script. If the variable is not set, the width is set to 80 columns. The minimum width is 40 columns. The maximum possible width is 160 columns.

This storage is used for storing data on XA transactions. On startup large enough to handle all the units of recovery for XA transactions from gateways in the same HA group and at runtime it needs to be large enough to handle the maximum number of active XA transactions.

Set the value of CTG_XA_MAX_TRAN to be 1000 or a value greater than the number of configured connection managers, which ever is the larger. For more information see Highly available Gateway group and Cold start."

CTG_XA_MAX_TRAN

. Set this environment variable to limit the maximum number of concurrent

XA transactions that a Gateway daemon can process. The range of values is 1 to 8192. The default value for a Gateway daemon is 1000.

CICS Transaction Gateway allocates an amount of additional storage on startup approximately equivalent to 200 times the value of `CTG_XA_MAX_TRAN` KB. For example, if the value of `CTG_XA_MAX_TRAN` is 1024, then 200 KB of additional storage is allocated on startup. This storage is used for storing data on XA transactions. On startup, the storage must be large enough to handle all the units of recovery for XA transactions from gateways in the same HA group and at runtime it needs to be large enough to handle the maximum number of active XA transactions.

Set the value of CTG_XA_MAX_TRAN to be 1000 or a value greater than the number of configured connection managers, which ever is the larger. For more information see "Highly available Gateway group" on page 76 and "Cold start" on page 245.

CTGENVVAR

Specifies the fully qualified location of a ctgenvvar script to be invoked by ctgstart. The script is self-contained and exports environment variables relevant to the Gateway daemon and associated CICS servers.

If CTGENVVAR is not defined, the ctgstart script looks for the ctgenvvar script in the product bin directory (for compatibility with previous versions). If no script is found in either location, ctgstart still attempts to initialize the Gateway daemon, on the assumption that the required environment variables are already in place.

Use this environment variable when starting the Gateway daemon from the USS shell, rather than through CTGBATCH.

CTGSTART_OPTS

Specifies options that are too long for inclusion in the JCL step. Because the PARM string used by the EXEC PGM= step in JCL is limited to 100 characters, there is not enough space to include some of the longer Gateway daemon startup options (for example, JVM properties for enabling trace and setting the trace file). Use this environment variable to specify those options too long for inclusion in the JCL step. The ctgstart script recognizes this environment variable and appends the supplied options to the Gateway daemon startup command. The value of the environment variable must not exceed 300 characters. For example, the code to turn on CICS TG tracing for Gateway daemon initialization is as follows:

```
CTGSTART_OPTS=-j-Dgateway.T=on
```

PATH Specifies the path in HFS containing the runtime resources necessary to run ctgstart. These are typically found in `/bin`, and the Java `bin` runtime directory.

TMPDIR

Specifies a temporary directory other than `/tmp`.

TZ Specify the local time zone and daylight saving time.

The value in this field is similar to the setting of TZ in /etc/profile. The full format is:

```
TZ=standardHH[:MM[:SS]] [daylight[HH[:MM[:SS:]]]
[,startdate[/starttime],enddate[/endtime]]]
```

An example for the United Kingdom is `TZ=GMT0BST,M3.5.0,M10.4.0`. An example for United States Eastern Standard time is `TZ=EST5EDT`

For further information on time zones, see *z/OS UNIX System Services Command Reference*.

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 CICSplex, and for subsequent task correlation in CICS.

Gateway identification has implications for the following:
- Qualification of log messages
- User interaction via z/OS console
- RRS registration for extended mode LUW or XA transaction support
- IPIC connections to identify the Gateway daemon to CICS
- Gateway daemon statistics and SMF records
- 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 “Gateway identity considerations” and “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 107.

Request monitoring data

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

Gateway identity considerations

The configuration of your Gateway daemon directly influences the choices available for Gateway identity the rules are detailed in this section.

When configuring a Gateway daemon there are three possibilities to consider for the APPLID qualifier and APPLID parameters.
- Define both APPLID qualifier and APPLID to form a fully-qualified APPLID
- Define only an APPLID
- Do not define either APPLID qualifier or APPLID

APPLID qualifier and APPLID

You must define both APPLID qualifier and APPLID when:
- XA transaction support is enabled (xasupport=on)

- Multiple Gateway daemons are organized into High availability Gateway groups for purposes of scalability with TCP/IP load balancing, or for administrative purposes

The Gateway daemon uses the APPLID for log messages, Gateway daemon statistics and SMF records. The JOBNAME is used for systems management with the **`MODIFY (/F)`** command.

The fully-qualified APPLID is used to register the Gateway daemon with RRS using the following structured resource manager name:

```
CICSTG.<APPLID qualifier>.<APPLID>[.UA]
```

If XA transaction support is not enabled, the resource manager name is appended with ".UA".

The fully-qualified APPLID is used to identify the Gateway daemon to CICS servers connected using the IPIC protocol.

APPLID only

You can define an APPLID when the following criteria are met:

- XA transaction support is disabled (xasupport=off)
- The Gateway daemon is registered with RRS for Extended LUW transaction support using a resource manager name defined by an environment variable CTG_RRMNAME, or generated by the Gateway daemon during initialization

The Gateway daemon uses the specified APPLID for log messages, Gateway daemon statistics and SMF records. The JOBNAME is used for systems management with the **`MODIFY (/F)`** command.

If the deprecated environment variable *CTG_RRMNAME* is defined, it is used for RRS registration, otherwise the following structured resource manager name is used:

```
CICSTG.<APPLID>.UA
```

The APPLID is used to identify the Gateway daemon to CICS servers connected using the IPIC protocol.

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 (**`applidqualifier`**).

APPLID undefined

You can leave the APPLID and qualifier undefined if XA transaction support is disabled (xasupport=off). In this situation the Gateway uses the JOBNAME as the identifier in log messages and SMF records.

Uniqueness

The defined APPLID qualifier or APPLID are not explicitly verified for uniqueness during initialization. If they are not unique within the sysplex or your wider organization, failures might occur during initialization or later when connecting to a CICS server.

It is recommended that you choose a naming convention that gives each Gateway daemon an APPLID that is unique within the sysplex or Gateway group and a fully-qualified APPLID that is unique within the wider organization.

When the fully-qualified APPLID is used for RRS registration the Gateway fails to initialize if another address space, usually a Gateway daemon, within the sysplex is already registered using the same resource name.

You might encounter connection problems if more than one Gateway attempts to establish an IPIC connection to a CICS server when both have the same APPLID or fully-qualified APPLID. The Gateway daemons might not be within the same sysplex or might be connecting from a different platform.

Related concepts:

"TCP/IP load balancing" on page 73
Load balancing ensures high system availability through the distribution of workload across multiple components.

Gateway APPLID

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

`applid=<name>`

Description

Set a value of up to eight characters. The value must be unique within the 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. 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.

The **`applid`** and **`applidqualifier`** parameters can be overridden on the *ctgstart* command. For more information, see "Options on the ctgstart command" on page 254

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

Default value

There is no default value for this parameter.

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.

The **`applid`** and **`applidqualifier`** parameters can be overridden on the *ctgstart* command. For more information, see "Options on the ctgstart command" on page 254

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.

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 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 107.

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.

EXCI server connections

For EXCI SYNCONRETURN requests through the Gateway daemon, the Gateway fully qualified APPLID is used to create the EXCI network UOWID and in local mode the Client application fully qualified APPLID is used.

SMF

If an APPLID is specified in the configuration file, it is written to the relevant field in the SMF record header. If no APPLID is specified, the job name is written to the APPLID field in the SMF record header. SMF does not use the APPLID qualifier.

z/OS console

If an APPLID is specified in the configuration file this is the message format.

```
CTGnnnnI applid <TEXT>
```

If no APPLID is specified, the message format is:

```
CTGnnnnI jobname <TEXT>
```

Configuring CICS server connections

The connections that CICS Transaction Gateway uses when forwarding client requests to CICS must be configured for the supported protocol.

EXCI and IPIC are the communications interfaces for CICS Transaction Gateway for z/OS. If you are using IPIC exclusively, you do not need to configure EXCI. Use the CTG_EXCI_INIT environment variable to specify that EXCI is not loaded.

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.

Transactional support

IPIC supports two-phase commit XA transactions in both local and remote modes.

For information about the transaction types supported by the IPIC protocol when using CICS Transaction Gateway to connect to different CICS servers see "Which protocol can be used?" on page 14

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 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:

"Which API can be used?" on page 14
This table shows which APIs are supported over the IPIC and EXCI protocols in local and remote mode.

"Which protocol can be used?" on page 14
This table shows what support is available for connecting to different version CICS servers over IPIC and EXCI.

"IPIC connection security" on page 43
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.

Configuring IPIC on CICS Transaction Server for z/OS

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

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

1. Set the System Initialization (SIT) parameter TCPIP=YES.
2. Define the TCP/IP address and host name for the z/OS system. By default, they are defined in the PROFILE.TCPIP and TCPIP.DATA data sets.

3. Add a TCP/IP listener to CICS. Use the following CEDA command to define a TCPIPSERVICE in a group:

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

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

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

   PRotocol
   The protocol of the service is IPIC.

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

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

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

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

   ```
   CEDA INS TCPIPSERVICE(service-name) GROUP(group-name)
   ```
5. Choose whether to predefine or to autoinstall IPIC connections in CICS Transaction Server for z/OS. Specific inbound connections can be defined for different configurations using the CICS definition, IPCONN, or the connection can be autoinstalled using either the default or a customized autoinstall program. When CICS TG connects to CICS it flows the fully-qualified APPLID defined for the Gateway daemon or local mode application and if this matches that defined on an IPCONN definition, that definition is used to install the connection. If there is no matching IPCONN definition, the connection is autoinstalled. For further information on setting the fully-qualified APPLID for IPIC connections see “IPIC server connections” on page 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 43.

   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 5. 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 161.

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. For more information, see "ECI resource adapter deployment parameters" on page 170. If you use a Java application, you can change this value by setting the LOCAL_PROP_IPIC_SENDSESSIONS property on the JavaGateway object. For more information, see CICS® Transaction Gateway Base API Programming Reference v9.0.0.1 .

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 CICS Transaction Gateway uses EXCI to attempt to connect to the CICS server.

Configuring an IPIC CICS Server definition:

To configure a new IPIC CICS server definition edit the IPICSERVER section of the configuration file.

The combination of fields that identify an IPIC server are Hostname or IP address, Port, CICS APPLID and CICS APPLID qualifier. This combination must be unique for each IPIC server definition to ensure that every connection that the CICS server and the network protocol see is represented by a unique server definition in the configuration file.

To configure an IPIC server definition edit the configuration file, see "IPICSERVER section of the configuration file" on page 161 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 161.

Default value

There is no default value for this parameter.

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 161 and the .

Default value
There is no default value for this parameter.

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 161.

Default value
There is no default value for this parameter.

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 161.

Default value
There is no default value for this parameter.

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 161.

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

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 161.

Default value
There is no default value for this parameter.

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 161.

Default value
There is no default value for this parameter.

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. 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 161.

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

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 161.

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

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 161.

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

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 161.

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

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 161.

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

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.

Configuring EXCI

The configuration of EXCI server connections is the same for both local and remote modes. A server definition is not required in the configuration file (ctg.ini) for remote mode. EXCI configuration is achieved through setting of environment variables.

Setting EXCI environment variables

If you are using the EXCI interface, you must ensure that the EXCI environment variables have been set correctly.

The EXCI environment variables are:

CTG_EXCI_INIT
Specifies whether or not EXCI is loaded.

CTG_PIPE_REUSE
Specifies how allocated EXCI pipes are reused.

DFHJVPIPE
Specifies the name of the pipe that CICS Transaction Gateway uses for EXCI calls.

DFHJVSYSTEM_nn
Specifies the name and description of an EXCI connected CICS server to be returned in response to a request for the CICS_EciListSystems function. The first system should be numbered 00.

STEPLIB
Specifies the library containing the default EXCI options and the EXCI load modules.

For more information about EXCI environment variables see “Environment variables: local and remote mode” on page 100.

You must also ensure that any customized EXCI options are used at startup. For more information see “Referencing your customized EXCI options” on page 121.

Customizing EXCI options

The EXCI options table, DFHXCOPT, allows you to specify a number of parameters for the EXCI. There is no suffixed version of this program, so the first DFHXCOPT table located in the STEPLIB concatenation is loaded. If you are using IPIC, you do not need to customize EXCI.

The following table shows the default parameters for the DFHXCOPT macro.

```
DFHXCO TYPE=CSECT,
        TIMEOUT=0,            No timeout
        TRACE=OFF,            Only Exception trace entries
        TRACESZE=16,          16K trace table
        DURETRY=30,           Retry SDUMPS for 30 seconds
        TRAP=OFF,            DFHXCTRA - OFF
        GTF=OFF,             GTF - OFF
        MSGCASE=MIXED,        Mixed case messages
        CICSSVC=0,            EXCI will obtain CICS SVC number
        CONFDATA=SHOW,        Show user commarea data in trace
        SURROGCHK=YES         Perform surrogate-user check @P1C
     END DFHXCOPT
```

Figure 11. EXCI options table DFHXCOPT

For full details see the *CICS Transaction Server for z/OS CICS External Interfaces Guide*. The relevant parameters, shown in bold, are:

TRACE
The default value OFF means that EXCI tracing is not required. Exception trace entries are always written to the internal trace table.

GTF
If you want to copy entries in the EXCI trace table to the generalized trace facility (GTF), specify GTF=ON.

CICSSVC
This specifies the CICS type 3 SVC number being used for MRO communication. EXCI must use the same SVC number that is used by the CICS MRO servers that are in the z/OS image where the client program is running. If you do not specify a specific CICS SVC number, the external CICS interface determines the SVC in use for MRO by means of an z/OS VERIFY command.

The default value zero means that EXCI will get the CICS SVC number from z/OS. Specify 0 only when you are sure that at least one CICS server has logged on to DFHIRP during the life of the z/OS IPL. If you use the default, and the external CICS interface requests the SVC from z/OS, the request will fail if no CICS server has logged on to DFHIRP.

In other circumstances, specify the CICS SVC number, in the range 200–255, that is in use for CICS interregion communications. This must be the SVC number that is installed in the z/OS image in which the client program is running (the local z/OS).

Note: All CICS servers using MRO within the same z/OS image must use the highest level of both DFHIRP and the CICS SVC, DFHCSVC. If your MRO CICSplex consists of CICS servers at different release levels, the DFHIRP and DFHCSVC installed in the LPA must be from highest release level of CICS within the CICSplex.

CONFDATA
The default value SHOW allows user data to be traced. If you wish to hide it, set this value to HIDETC.

SURROGCHK
The default value YES means that a surrogate security check will be performed in the EXCI client address space if the flowed user ID is different from the user ID of the client address space. This occurs regardless of the CICS security settings. This means that if you flow a user ID in an EXCI request (including

flowing a user ID with an External Call Interface [ECI] request using the CICS Transaction Gateway for z/OS), you need either to add an RACF surrogate profile, or disable surrogate security checks by setting this parameter to NO.

Note: If the CICS server has no security (SEC=NO), specify SURROGCHK=NO. For more information, see the *CICS Transaction Server for z/OS RACF Security Guide*.

Referencing your customized EXCI options:

The EXCI options environment variable must point to the library that contains the table of EXCI customization options. This ensures that these options are used at startup.

Using STDENV

The STEPLIB environment variable should be updated to reference the dataset containing your customized DFHXCOPT table ahead of the CICS EXCI load library. For more information, see "Environment variables: local and remote mode" on page 100

Using ctgenvvars

If you are using the `ctgenvvar` script you should update the EXCI_OPTIONS environment variable to specify the library that contains the customized options table. Edit the `ctgenvvar` script. Find the line:

```
EXCI_OPTIONS="your.user.loadlib"
```

and change it to specify the appropriate library.

You must update the name of the library specified in the **`EXCI_LOADLIB`** environment variable to that of the latest version of CICS being used. This library contains:
- The default EXCI options
- The EXCI load modules

Update **`EXCI_LOADLIB`** by editing the `ctgenvvar` script. The sample `ctgenvvar` script contains the following definition:

```
# Set EXCI_LOADLIB to the CICS SDFHEXCI library.
EXCI_LOADLIB="<EXCI_LOADLIB>"
```

Change it to specify the appropriate library and high-level qualifier. The **`EXCI_OPTIONS`** and **`EXCI_LOADLIB`** variables are used in the definition of the **`STEPLIB`** environment variable.

For more information about the EXCI options and how to customize them, see the *CICS Transaction Server for z/OS CICS External Interfaces Guide*.

Configuring EXCI on CICS Transaction Server for z/OS

This information describes how to configure EXCI connections to allow the CICS Transaction Gateway to connect to a CICS Transaction Server. See the *CICS Transaction Server for z/OS Installation Guide* for further details on EXCI configuration.

CICS server connection definition:

CICS Transaction Server includes sample EXCI connection and session definitions (EXCG and EXCS). These are in the sample group DFH$EXCI.

You can use the supplied definitions without modification, you can copy and modify the supplied definitions, or you can create completely new definitions.

Here are two examples of a connection definition:

```
VIEW GROUP(MYEXCI) CONNECTION(JCOS)
 OBJECT CHARACTERISTICS                                CICS RELEASE = 0620
  CEDA  View Connection( JCOS )
   Connection     : JCOS
   Group          : MYEXCI
   DEscription    : Sample EXCI Specific connection
  CONNECTION IDENTIFIERS
   Netname        : JGATE400
   INDsys         :
  REMOTE ATTRIBUTES
   REMOTESYSTem   :
   REMOTEName     :
   REMOTESYSNet   :
  CONNECTION PROPERTIES
   ACcessmethod   : IRc                Vtam | IRc | INdirect | Xm
   PRotocol       : Exci               Appc | Lu61 | Exci
   Conntype       : Specific           Generic | Specific
   SInglesess     : No                 No | Yes
   DAtastream     : User               User | 3270 | SCs | STrfield | Lms
+  RECordformat   : U                  U | Vb

                                                      SYSID=CW2C APPLID=IYCWZCFY

PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

```
 OBJECT CHARACTERISTICS                                CICS RELEASE = 0620
  CEDA  View Connection( JCOS )
   Queuelimit     : No                 No | 0-9999
   Maxqtime       : No                 No | 0-9999
  OPERATIONAL PROPERTIES
   AUtoconnect    : No                 No | Yes | All
   INService      : Yes                Yes | No
  SECURITY
   SEcurityname   :
   ATtachsec      : Identify           Local | Identify | Verify | Persistent | Mixidpe
   BINDPassword   :                    PASSWORD NOT SPECIFIED
   BINDSecurity   : No                 No | Yes
   Usedfltuser    : No                 No | Yes
  RECOVERY
   PSrecovery     :                    Sysdefault | None
   Xlnaction      : Keep               Keep | Force




                                                      SYSID=CW2C APPLID=IYCWZCFY

PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

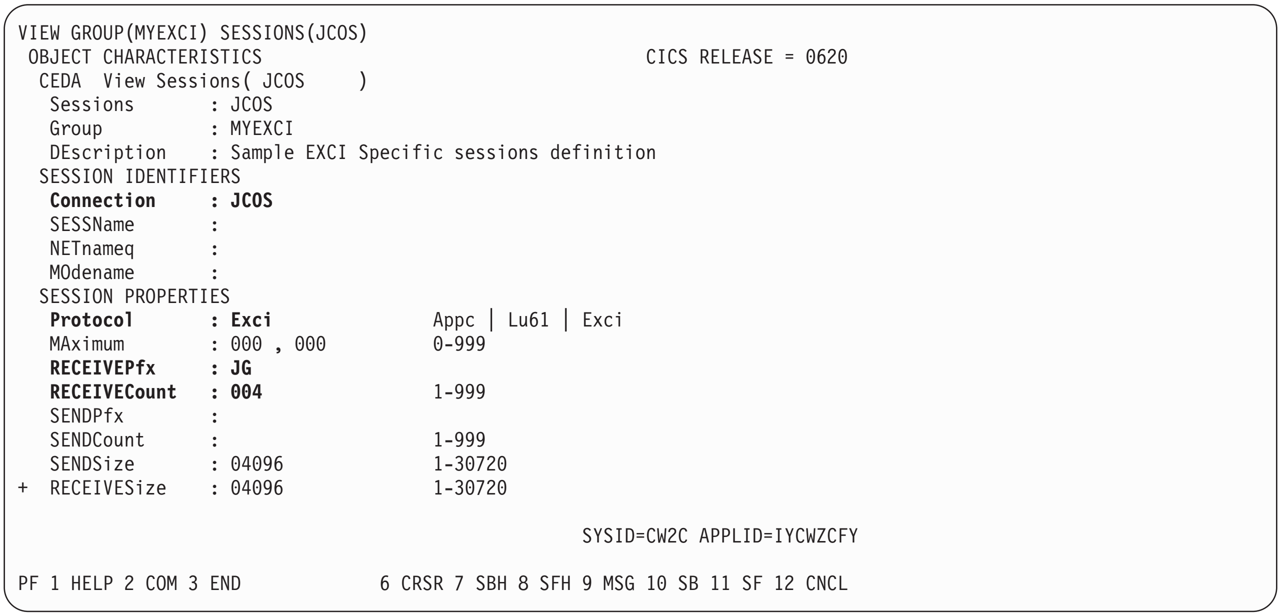
The key parameters are shown in bold in the two examples, these are:

Parameter	Setting
`Netname`	If you enter a Netname, the connection uses a specific pipe. If you leave it blank, the EXCI connection uses a generic pipe. You should ideally use a specific pipe because this helps to manage multiple connections, and to identify problems. If you use specific pipes, set Netname to be the same as the pipe specified in the EXCI client program. You specify this on the user_name parameter of an Initialize_User EXCI call. Set the DFHJVPIPE environment variable to allow CICS Transaction Gateway to pass this value to the EXCI. For more information see Setting environment variables. If you use the sample DFH$EXCI, set the DFHJVPIPE environment variable to BATCHCLI to use the specific pipe. If CICS Transaction Gateway is configured to connect to more than one CICS server, use the same Netname for all EXCI connections.
`ACcessmethod`	Specify **`IRC`** (interregion communication).
`PRotocol`	Specify **`EXCI`**.
`Conntype`	This defines the type of EXCI connection used for jobs that are not CICS to communicate with a CICS server on z/OS. This is a type of multiregion operation (MRO) request and can be either Generic or Specific. • Generic: An MRO link with multiple sessions to be shared by multiple EXCI users. Only one generic EXCI connection can be installed in each server. Specify Generic if you left the Netname attribute blank. • Specific: An MRO link with one or more sessions dedicated to a single user in a client program. Specify Specific if you completed the Netname attribute.
`ATtachsec`	This determines whether a check is to be made against the flowed user ID. Specify Local or Identify. • Local: CICS Transaction Gateway does not flow a user identifier; just the link user ID (if specified) is used. If no link user ID is supplied, all requests are run under the CICS default user ID as specified in the DFLTUSER SIT parameter. • Identify: A user ID is flowed on every request, but a password is not expected, because CICS trusts the user ID as having been already authenticated. For security reasons, consider enabling user ID and password verification within CICS Transaction Gateway, before the EXCI request is made. The user ID is either the user named in the ECIRequest object(If null), or the user ID of the thread under which the ECI request runs.

For more information on how to create CICS server connection definitions see the *CICS Transaction Server for z/OS External Interfaces Guide*.

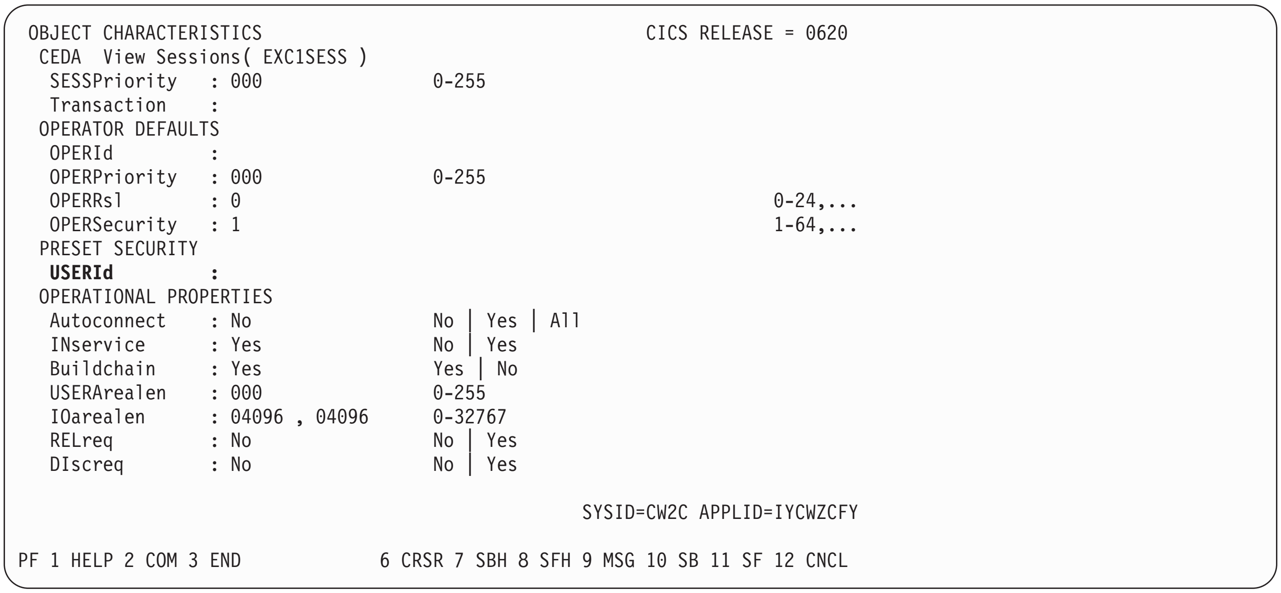
CICS server sessions definition:

For each EXCI connection definition, you create a sessions definition.



```
VIEW GROUP(MYEXCI) SESSIONS(JCOS)
 OBJECT CHARACTERISTICS                                    CICS RELEASE = 0620
  CEDA  View Sessions( JCOS     )
   Sessions       : JCOS
   Group          : MYEXCI
   DEscription    : Sample EXCI Specific sessions definition
  SESSION IDENTIFIERS
   Connection     : JCOS
   SESSName       :
   NETnameq       :
   MOdename       :
  SESSION PROPERTIES
   Protocol       : Exci               Appc | Lu61 | Exci
   MAximum        : 000 , 000          0-999
   RECEIVEPfx     : JG
   RECEIVECount   : 004                1-999
   SENDPfx        :
   SENDCount      :                    1-999
   SENDSize       : 04096              1-30720
+  RECEIVESize    : 04096              1-30720

                                                     SYSID=CW2C APPLID=IYCWZCFY

PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 12. Sessions definition—screen 1



```
 OBJECT CHARACTERISTICS                                    CICS RELEASE = 0620
  CEDA  View Sessions( EXC1SESS )
   SESSPriority   : 000                0-255
   Transaction    :
  OPERATOR DEFAULTS
   OPERId         :
   OPERPriority   : 000                0-255
   OPERRsl        : 0                                           0-24,...
   OPERSecurity   : 1                                           1-64,...
  PRESET SECURITY
   USERId         :
  OPERATIONAL PROPERTIES
   Autoconnect    : No                 No | Yes | All
   INservice      : Yes                No | Yes
   Buildchain     : Yes                Yes | No
   USERArealen    : 000                0-255
   IOarealen      : 04096 , 04096      0-32767
   RELreq         : No                 No | Yes
   DIscreq        : No                 No | Yes

                                                     SYSID=CW2C APPLID=IYCWZCFY

PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 13. Sessions definition—screen 2

The relevant parameters (shown in bold in the figures) are:

`Connection`
This sessions definition is associated with the JCOS connection.

`Protocol`
Specify **EXCI**.

`RECEIVEPfx`
This is a one or two character prefix for the receive session names. CICS creates the remainder of the four character names from the alphanumeric characters A through Z, and 1 through 9. These two or three character identifiers begin with the letters AAA, and continue in ascending sequence

until the number of session entries reaches the limit set by the RECEIVECount value (in our example JGAA, JGAB, JGAC, and so on). The default receive prefix is the < symbol.

RECEIVECount
This defines the number of sessions that can receive before sending. Because all EXCI sessions are receive sessions, this defines the number of pipes that can be used simultaneously. Because the EXCI imposes a maximum of LOGONLIM pipes for each EXCI address space, set RECEIVECount to a value larger than LOGONLIM.

SENDCount
Leave blank. Because EXCI sessions can only receive, there are no send sessions.

USERId
This defines the preset user identifier to be used for link security checking. It provides an additional security check for each transaction in addition to the flowed user ID. If you do not specify a preset user ID for link security, CICS uses the flowed user ID.

Open interregion communication:

After you have installed the CONNECTION definitions, you must set interregion communication (IRC) open.

```
CEDA INSTALL GROUP(groupname)
```

to install the group containing the definitions, and:

```
CEMT SET IRC OPEN
```

to set IRC open. For more information see the *CICS Resource Definition Guide.*

EXCI user-replaceable module DFHXCURM:

The user-replaceable module **DFHXCURM** is invoked on EXCI **Allocate_Pipe** requests, and also after detection of EXCI retryable errors.

This occurs in one of these circumstances:
- The target CICS server is not available.
- No pipes are available on the target CICS server.
- IRC has not been active since the last initial program load (IPL).

In early releases of CICS Transaction Gateway, an EXCI pipe was established, used, and deallocated, for each ECI flow. CICS Transaction Gateway now creates a pipe only for the first ECI flow. Later calls reuse this pipe. As a result, it is no longer possible to perform workload management using the EXCI user-replaceable module **DFHXCURM**, because the code is called only on the first ECI flow.

An alternative way of performing workload balancing is to take the following steps:
1. Route all your ECI flows to the same CICS server.
2. Perform the routing to other CICS servers, using, for example, a round-robin approach to distribution.

Although reusing the same pipe for EXCI calls has an impact on previously written workload management code, there are benefits in CICS Transaction

Gateway performance. This is because the **`Initialize_User`** and **`Allocate_Pipe`** EXCI commands are only called once per worker thread per APPLID. In the past they were called at the start of every ECI flow.

If an EXCI Open_Pipe call fails with a retryable error, the CICS Transaction Gateway retries the Allocate_Pipe and Open_Pipe calls up to a maximum of five times, (the same as the EXEC CICS LINK command.) Each time the Allocate_Pipe is called, the EXCI user-replaceable module DFHXCURM is driven and can change the specified CICS_APPLID; this allows the CICS Transaction Gateway five retries to find an available server.

An EXCI pipe to a failed CICS server is deallocated and reallocated when a subsequent ECI request to that failed CICS server is made. DFHXCURM can be driven to allocate the pipes to a different server, but the pipes are not reallocated to the failed server when it comes up again.

Configuring XA support

This information describes what you must do to configure your system for XA support.

Configuring for XA transaction support

You can configure CICS Transaction Gateway to benefit from XA transactionality. XA transactionality provides support for two-phase commit transactions and allows CICS Transaction Gateway to participate in global transactions. This section describes how to configure a single CICS Transaction Gateway and multiple CICS Transaction Gateways in a highly-available Gateway group.

Ensuring you have a working Gateway daemon configuration

Before you configure CICS Transaction Gateway with XA support you must have a working Gateway daemon configuration. If you do not have a working Gateway daemon configuration, complete the following tasks:

1. Install CICS Transaction Gateway.
2. Configure RACF authorization for user ID and password authentication. For more information see “Configuring for client certificate mapping” on page 143.

Configuring the application server and client environment with XA support

Deploy the CICS resource adapters as described in Deploying the CICS resource adapter.

- If your applications connect to CICS Transaction Gateway in remote mode, you can use the ECI resource adapter from any supported version of CICS Transaction Gateway.
- If you have any applications which connect to CICS Transaction Gateway in local mode, you must use the ECI resource adapter from the same release as your installed version of CICS Transaction Gateway.

Configuring the Gateway daemon with XA support

1. Ensure that CTGRRMS services are enabled as described in “Enabling CTGRRMS services” on page 127.
2. You must permit access for the Gateway daemon USERID to one of the following RACF facilities when XA support is enabled:

- ALTER access to the MVSADMIN.RRS.COMMANDS.** facility.
- ALTER access to the MVSADMIN.RRS.COMMANDS.gname.sysname facility.

  gname
  : is the logging group name and corresponds to the logging group in the RRS administrative panels in ISPF. Set **gname** to the value for the sysplex where the Gateway is running.

  sysname
  : is the system name. Set **sysname** to the value for the LPAR where the Gateway is running

  If you give ALTER access to MVSADMIN.RRS.COMMANDS.** the Gateway daemon is permitted to perform recovery operations for transactions associated with any system name or logging group. This option requires less administration but does not provide granularity of control.

  If you give ALTER access to MVSADMIN.RRS.COMMANDS.gname.sysname the Gateway daemon is permitted to perform recovery operations for transactions associated with the specified system name or logging group. This option allows for greater granularity of control but requires a greater amount of administration.

  If workload is shared between multiple Gateway daemons within a highly-available Gateway group, the Gateway USERID must have ALTER access to each **gname.sysname** pair used by the highly-available Gateway group.

3. Set the **`xasupport`** configuration keyword in the GATEWAY section of the configuration file (ctg.ini). For more information see “GATEWAY section of the configuration file” on page 158.
4. Decide on a fully-qualified APPLID for the CICS Transaction Gateway instance.

   If you use TCP/IP load balancing across multiple Gateway daemons in a highly-available Gateway group, you must specify the same APPLID qualifier for each Gateway daemon in the group; this denotes each individual Gateway daemon's membership of the group. Gateway daemons in a highly-available Gateway group can span multiple LPARs in a sysplex.

   See *Configuring identification using APPLID* for more information about setting a fully-qualified APPLID.

   The APPLID and APPLIDQUALIFIER are defined in the configuration file or as command line overrides.

   The Gateway daemon uses the following name format to register with RRS: `CICSTG.`*`APPLIDQUALIFIER.APPLID`*.
5. Start each Gateway daemon as described in “Starting CICS Transaction Gateway” on page 244.

See Sysplex restrictions for information about z/OS image restrictions.

Enabling CTGRRMS services

To support XA, CICS Transaction Gateway issues authorized RRS calls, through CTGRRMS services. Protect these services with RACF, or another external security manager. Follow these instructions to add the new services, and to protect them.

1. Add *hlq*.SCTGLINK to the MVS LNKLST. This load library must be APF-authorized.
2. Issue the MVS command F LLA,REFRESH to refresh the LNKLST LOOKASIDE address space (LLA).

3. Give the user ID that is running the CICS Transaction Gateway UPDATE authority to the RACF entity CTG.RRMS.SERVICE in the FACILITY class. Activate SETROPTS RACLIST processing for the FACILITY general resource class. When you activate this function, you improve performance because I/O to the RACF database is reduced. If you are using an external security manager other than RACF, see its documentation for information about how to give the user ID that is running CICS Transaction Gateway access to these resources. Issue the TSO commands:
   a.
      ```
      SETROPTS RACLIST(FACILITY)
      ```

      Note: If you activate SETROPTS RACLIST processing for the FACILITY class, any time you make a change to a FACILITY profile, you must also refresh SETROPTS RACLIST processing for the FACILITY class for the change to take effect (`SETROPTS RACLIST(FACILITY) REFRESH`).
   b. Create the entity in the FACILITY class:
      ```
      RDEFINE FACILITY CTG.RRMS.SERVICE UACC(NONE)
      ```
   c. Enable the user ID for the CICS Transaction Gateway (*gway_id*):
      ```
      PERMIT CTG.RRMS.SERVICE CLASS(FACILITY) gway_id ACCESS(UPDATE)
      ```

If the CTGINIT module in SCTGLINK is subsequently refreshed, see "Starting, stopping or refreshing the CTGRRMS services."

Starting, stopping or refreshing the CTGRRMS services

Use the <install_path>/bin/ctgasi command to start, stop or refresh the CTGRRMS services that are used for XA support.

There are three levels of access for the CTG.RRMS.SERVICE RACF entity in the FACILITY class. These levels control the tasks that can be performed. The levels are:

- `UPDATE` - this allows the user ID of a Gateway daemon to use the CTG.RRMS.SERVICE facility. This level of access is required for a Gateway daemon configured for XA support.
- `CONTROL` or `ALTER` - these access levels allow a user ID such as a System Programmer to perform shutdown or refresh operations on the CTGRRMS address space, with the <install_path>/bin/ctgasi utility.
- `NONE` or `READ` - these levels prevent a user ID from access to the CTGRRMS services.

Use the <install_path>/bin/ctgasi command to start, stop or refresh the CTGRRMS services that are used for XA support.

Starting CTGRRMS services

Use the <install_path>/bin/ctgasi command to start CTGRRMS services that are used for XA support. **`ctgasi`** must be run in a UNIX System Services shell. It can be launched in batch mode using CTGBATCH.

About this task

If the CTGRRMS address space is not already running when a Gateway daemon with XA support is started, then `ctgasi` is automatically invoked silently to start CTGRRMS. You can also manually start CTGRRMS by using the **`ctgasi`** command if, for example, CICS Transaction Gateway fails to start, or a new version of the CTGINIT module in SCTGLINK is distributed.

The user ID executing the Gateway daemon, or `ctgasi`, must have at least `UPDATE` level access to the CTG.RRMS.SERVICE in the facility class to start CTGRRMS.

Follow these steps after successfully installing CICS Transaction Gateway and before running any Gateway daemons:

Procedure

1. Ensure that the user ID which runs **`ctgasi`** has UPDATE, CONTROL, or ALTER authority to the RACF FACILITY CTG.RRMS.SERVICE and log on using this user ID.
2. Shut down all existing Gateway daemon address spaces, if present, in the logical partition (LPAR).
3. Add *hlq*.SCTGLINK to the MVS LNKLST. This load library must be APF-authorized.
4. Issue the MVS command:

   ```
   F LLA,REFRESH
   ```

   to refresh the LNKLST LOOKASIDE address space (LLA).
5. Issue the following command:

   ```
   ctgasi
   ```
6. See the following messages which are issued to the system log when CTGRRMS services are started successfully:

   ```
   CTG6241I Initializing CTGRRMS Services
   CTG6242I  CTGRRMS Services open for business
   ```
7. Repeat these steps on every LPAR on which you install this version of CICS Transaction Gateway.

Refreshing CTGRRMS services

Use the <install_path>/bin/ctgasi command to refresh CTGRRMS services that are used for XA support. **`ctgasi`** must be run in a UNIX System Services shell. It can be launched in batch mode using CTGBATCH.

About this task

CTGRRMS is a long-running task used by the XA logic in CICS Transaction Gateway. The executable code for the CTGRRMS services is in the CTGINIT module shipped with CICS Transaction Gateway. If the version of CTGINIT is incremented with a new release of CICS Transaction Gateway, you must restart CTGRRMS to use the new version of CTGINIT. If CTGINIT is at a version that is lower than the version of the Gateway daemon you have installed, the Gateway daemon will fail to startup when configured to use XA. CTGRRMS was introduced in Version 6.1 and upgraded at Version 7.2.

Follow these steps after successfully installing CICS Transaction Gateway and before running any Gateway daemons. If you do not follow these steps for every LPAR, Gateway daemons will fail to start.

Procedure

1. Ensure that the user ID which runs **`ctgasi`** has CONTROL authority to the RACF FACILITY CTG.RRMS.SERVICE and log on using this user ID.

2. When refreshing the CTGRRMS services, it is good practice to use a different user ID to the user ID used to run the Gateway daemon because a higher level of access to CTG.RRMS.SERVICE is required. Enable the refresh user ID (*rfreshid*):
   ```
   PERMIT CTG.RRMS.SERVICE CLASS(FACILITY) rfreshid ACCESS(CONTROL)
   ```
3. Shut down all existing Gateway daemons in the LPAR, and ctgmaster if you are using an earlier version of the CICS Transaction Gateway in this LPAR.
4. Add *hlq*.SCTGLINK to the MVS LNKLST. *hlq* is the install directory of the latest level of CICS Transaction Gateway installed on the LPAR. This load library must be APF-authorized.
5. Issue the MVS command:
   ```
   F LLA,REFRESH
   ```

   to refresh the LNKLST LOOKASIDE address space (LLA).
6. Issue the following commands in the following order:
   ```
   ctgasi -s
   ctgasi
   ```

   to shut down and restart the CTGRRMS service.
7. If the command detects that instances of the CICS Transaction Gateway are running, it issues a warning and does not shut down CRGRRMS services. Check again that no Gateway daemon services and no ctgmaster processes (if you are using an earlier version in this LPAR) are running.
8. Repeat these steps on every LPAR on which you install this version of CICS Transaction Gateway.

Results

You can still run Gateway daemons and ctgmaster processes from earlier releases in the same LPAR after you have completed the refresh process. A running CTGRRMS service is compatible with earlier releases.

A Gateway daemon, or version of ctgasi, from an earlier release cannot run a CTGINIT file from a later release. The version of ctgasi which invokes CTGINIT and causes the service to be refreshed must be from the later release.

Stopping CTGRRMS services

Use the <install_path>/bin/ctgasi command to stop CTGRRMS services that are used for XA support. **`ctgasi`** must be run in a UNIX System Services shell. It can be launched in batch mode using CTGBATCH.

About this task

If you need to need to re-IPL a logical partition (LPAR), you also need to stop CTGRRMS services. Follow these steps:

Procedure

1. Ensure that the user ID which runs **`ctgasi`** has CONTROL or ALTER authority to the RACF FACILITY CTG.RRMS.SERVICE and log on using this user ID.
2. Shut down all existing Gateway daemons in the LPAR, and ctgmaster if you are using an earlier version of the CICS Transaction Gateway in this LPAR.
3. Issue the following command:
   ```
   ctgasi -s
   ```

4. Repeat these steps on every LPAR on which you install this version of CICS Transaction Gateway.

LPAR IPLs in an XA environment

If you need to re-IPL an LPAR in an XA environment, follow this procedure.

Procedure

1. Stop all Gateway daemons and the CTGRRMS services in the LPAR. See "Stopping CTGRRMS services" on page 130 for details about how to do this.
2. Perform the standard procedures required for an IPL of an LPAR.
3. Immediately following the IPL, start the RRMS services at the correct level. See "Starting CTGRRMS services" on page 128 for details about how to do this.

ctgasi command syntax

The ctgasi command manually starts, stops or refreshes the CTGRRMS services used for XA support.

```
ctgasi [[-refresh|-shutdown [-force]]|[-tracedump]][-verbose]
ctgasi [[-r|-s [-f]]|[-td]] [-v]
ctgasi [-?]
```

Issue the command without parameters to start an address space if one is not already running. The meaning of the optional parameters is as follows:

`-refresh | -r`
Stops the CTGRRMS services and then starts a new instance.

`-shutdown | -s`
Stops the CTGRRMS services, but does not start a new instance.

`-force | -f`
Forces the command to run even if active instances of the CICS Transaction Gateway are detected. Used with the `-refresh` or `-shutdown` parameters.

`-tracedump | -td`
Dumps a formatted initialization trace.

`-verbose | -v`
Displays extended messages while ctgasi is running.

`-?` Displays help on the command.

XA transaction support activate (`xasupport`)

The **`xasupport`** parameter controls whether the CICS Transaction Gateway supports XA transactions.

`xasupport=<on|off>`

Description
You can set the **`xasupport`** configuration file parameter to `on`, to enable support for XA based JEE global transactions.

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

Default value
The default value is `off`.

Gateway start type

The **`start`** parameter determines whether heuristically-completed transactions are resolved.

`start=<cold>`

Description

Set the value to **cold** to resolve heuristically-completed transactions. A heuristic error might occur after the failure of a transactional component or through manual intervention by an operator using the RRS panels, for more information, see “Cold start” on page 245.

You can use the **`ctgstart -start=`***cold* command to override the value of **`start`**. For more information, see “Options on the ctgstart command” on page 254.

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

Default value

By default, this parameter is not included in the configuration file.

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's RACF key ring or keystore.

Tools for working with digital certificates

Use these tools to work with digital certificates in different scenarios:

- Use keytool for software encryption, if the key ring is stored in zFS

- Use hwkeytool for hardware encryption, if the key ring is stored in zFS
- You can also use RACF for creating and maintaining certificates and key rings

Related information:

"Using keytool for certificate management"
The keytool command line application is provided with the SDK.

"Creating and maintaining hardware key ring files" on page 140
You can use the `hwkeytool` command that is provided as part of the IBM Java software development kit in much the same way as the `keytool` command to generate key rings and manage certificates. Extra parameters are available to specify how the key is stored on the cryptographic device, and how it is to be used. You also have the option of labeling the key on the cryptographic device.

"Using RACF key rings" on page 139
The key rings that CICS Transaction Gateway uses when establishing secure SSL connections are stored in RACF. This provides an alternative to Java keystore (.jks) files stored in the ZFS (a USS filesystem).

Using keytool for certificate management

The keytool command line application is provided with the 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:

Each *strvalue* is a string value. The meaning of the abbreviations is as follows:

- cn = common name
- o = organization
- ou = organization unit

- l = city/locality
- s = state/province
- c = country name

An example of an X.500 distinguished name is shown here:

```
"cn=someserver.location.ibm.com,o=IBM,ou=IBMGB,
   l=Winchester,s=Hants,c=GB"
```

Figure 14. An X.500 distinguished name

-keystore *location*
The key ring file location. For example: `ktserverss.jks`

-keypass *password*
The password used to protect the private key. Set this to the same value as the `-storepass` password, to enable the CICS Transaction Gateway to establish a connection over SSL.

-storepass *password*
The password used to protect the integrity of the key ring. Set this to the same value as the `-keypass` password, to enable the CICS Transaction Gateway to establish a connection over SSL.

-keyalg *algorithm*
The algorithm to be used to generate the key pair.

An example of this command is shown here:

```
keytool -genkey -alias exampleServerCert -keysize 1024
   -dname "cn=vmware2.location.ibm.com,o=IBM,ou=IBMGB,l=Winchester,s=Hants,c=GB"
   -keystore ktserverss.jks -keypass default -storepass default
   -keyalg RSA
```

Figure 15. 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 16. Using the keytool command to view certificates

Export the server's signer certificatd

The next step is to export the signer certificate and store it in a safe place. This can then be imported into the repository of any client that needs to connect to this SSL server.

The certificate is exported by using the following instance of the keytool command:

```
keytool -export -alias aliasname -keystore location
   -storepass password -file filename -rfc
```

Where the options are:

-export
Export a certificate.

-alias *aliasname*
Name of the key (in the key ring) to export.

-keystore *location*
The key ring location.

-storepass *password*
The password used to protect the integrity of the key ring.

-file *filename*
The name of the file to export the certificate to.

-rfc Export the certificate in RFC format (Base64 encoded ASCII).

An example of the keytool command to export a signer certificate is shown here:

```
keytool -export -alias exampleServerCert -keystore ktserverss.jks -storepass default
   -file exampleServerCertKT.arm -rfc
```

Figure 17. 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 ASCII mode.

Configuring your SSL clients

To configure your SSL clients you create a client key ring and import the server's signer certificate, create a self-signed certificate in the client. Next you export the client's signer certificate, and transfer the server certificate to the client. Finally you import the client signer certificate into the server's key ring file.

If your server does not use client authentication you complete the first task (create a client key ring and import the server's signer certificate) but you do not have to complete the other tasks.

Create a client key ring and import the server's signer certificate.

Issuing the following command to create the key ring and import the certificate:

```
keytool -import -alias aliasname -file certfile -keystore keystorefile
   -storepass password -noprompt
```

Where the options are:

-import
Import a certificate.

-alias *aliasname*
The name under which the certificate is to be stored.

-file *certfile*
The file that contains the certificate.

-keystore *keystorefile*
The key ring into which the certificate is to be imported.

-storepass *password*
The password used to protect the integrity of the key ring.

-noprompt
Removes the need to confirm that the certificate is imported.

An example of this command is shown here:

```
keytool -import -alias exampleServer -file exampleServerCertKT.arm -keystore clientStore.jks
   -storepass default -noprompt
```

Figure 18. Using the keytool command to create a key ring containing the server's signer certificate

Create a self-signed certificate in the client key ring

To create a new keystore containing a self-signed certificate use the following instance of the keytool command:

```
keytool -genkey -alias aliasname -keysize numericvalue -dname distname
   -keystore location -keypass password -storepass password
   -keyalg algorithm
```

The options are:

-genkey
Generates a key pair and wraps the public key into a self-signed certificate.

-alias *aliasname*
Defines the alias name that identifies the store containing the self-signed certificate and private key.

-keysize *numericvalue*
Defines the size of the key.

-dname *distname*
Specifies the X.500 distinguished name to be associated with the alias. This is used as the issuer and subject fields of the self-signed certificate. The distinguished name consists of a number of fields separated by commas in the following format:

Each `strvalue` is a string value. The meaning of the abbreviations is as follows:

- cn = common name
- o = organization
- ou = organization unit
- l = city/locality
- s = state/province
- c = country name

An example of an X.500 distinguished name is shown here:

```
"cn=someserver.location.ibm.com,o=IBM,ou=IBMGB,
   l=Winchester,s=Hants,c=GB"
```

Figure 19. An X.500 distinguished name

-keystore *location*
: The key ring file location. For example: `ktserverss.jks`

-keypass *password*
: The password used to protect the private key. Set this to the same value as the `-storepass` password, to enable the CICS Transaction Gateway to establish a connection over SSL.

-storepass *password*
: The password used to protect the integrity of the key ring. Set this to the same value as the `-keypass` password, to enable the CICS Transaction Gateway to establish a connection over SSL.

-keyalg *algorithm*
: The algorithm to be used to generate the key pair.

An example of the keytool command is shown here:

```
keytool -genkey -alias exampleClientCert -keysize 1024
   -dname "cn=John Doe,o=IBM,ou=IBMGB,l=Winchester,s=Hants,c=GB"
   -keystore clientStore.jks -keypass default -storepass default
   -keyalg RSA
```

Figure 20. 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 21. 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 ASCII 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. 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 **ctgstart -keyring=***file* 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.

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 **ctgstart -keyringpw=***password*, 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 unless using an ESM key ring.

Key ring password encryption

The **keyringpwscrambled** parameter specifies whether the **keyringpw** parameter value was encrypted.

keyringpwscrambled=<on|off>

Description

This value is set to **on** when the value of **`keyringpw`** is encrypted.

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

The Password Scrambler utility can be used to generate a scrambled keyring password definition suitable for putting in the configuration file. See the sample JCL in the `CTGSSLPW` member of the `samples` PDS for an example of how to run the password scrambler utility.

Default value

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

ESM key ring

The **`esmkeyring`** parameter takes the value of either `on` or `off` which indicates whether the key ring value is an external security manager (ESM) key ring or a keystore name.

`esmkeyring=<on|off>`

Description

Specify **`esmkeyring=on`** in the configuration file if you have a key ring provided by an ESM such as RACF.

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

Default value

The default value is `off`.

Use hardware cryptography

The **`hwcrypt`** parameter defines that the key ring is stored using hardware cryptographic functions.

`hwcrypt=<on|off>`

Description

Specify the **`hwcrypt=on`** parameter in the configuration file to enable hardware cryptography. Hardware cryptography is managed using an Integrated Cryptographic Services Facility (ICSF) card and associated software on the mainframe.

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

Default value

The default is **`off`**.

Using RACF key rings

The key rings that CICS Transaction Gateway uses when establishing secure SSL connections are stored in RACF. This provides an alternative to Java keystore (.jks) files stored in the ZFS (a USS filesystem).

Creating and maintaining key rings

The key ring must contain a personal certificate and the certificate authority certificate used to sign it. The key ring must be accessible by the user ID under which the Gateway daemon is running.

To create and maintain RACF key rings, you can either use the RACDCERT native command or the DIGITAL CERTIFICATES AND KEY RINGS panels found under the main RACF service options panel in ISPF.

For information on creating certificates and key rings in RACF, see the *z/OS Security Server RACF Security Administrator's Guide*.

Exporting certificates

The key ring that CICS Transaction Gateway uses must contain the personal certificate with its private key connected as a personal certificate. It must also contain the Certificate Authority certificate used to sign the personal certificate, attached as a CERTAUTH certificate. The use of certificates connected as SITE is not supported.

You export the personal certificate to the client keystore using FTP:
- If you export as FORMAT(CERTB64), you must FTP the file in ASCII format.
- If you export the certificate as FORMAT(CERTDER), you must FTP the file in binary format.

Defining key rings in the configuration file

To set the RACF key ring in the configuration file:
- Define the keyring entry as the name of the RACF key ring (omitting the keyringpw entry).
- Define the **`esmkeyring`** parameter. **`esmkeyring`** is specified in the PRODUCT section of the `ctg.ini` file.

For more information see "SSL protocol parameters" on page 160.

Creating and maintaining hardware key ring files

You can use the `hwkeytool` command that is provided as part of the IBM Java software development kit in much the same way as the `keytool` command to generate key rings and manage certificates. Extra parameters are available to specify how the key is stored on the cryptographic device, and how it is to be used. You also have the option of labeling the key on the cryptographic device.

To create a key ring, issue a command like the following:

```
hwkeytool -genkey -alias aliasname -keyalg algorithm
-storetype JCE4758KS -dname distname –keypass password
-storepass password -hardwaretype type -hardwareusage KEYMANAGEMENT
```

The options are as follows:

-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.

-keyalg *algorithm*
The algorithm to be used to generate the key pair. See your Java SDK documentation for details.

-storetype
The format of the keystore.

-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:

```
"cn=strvalue1,o=strvalue2,ou=strvalue3,
l=strvalue4,s=strvalue5,c=strvalue6"
```

Each `strvalue` is a string value. The meaning of the abbreviations is as follows:

- cn = common name
- o = organization
- ou = organization unit
- l = city/locality
- s = state/province
- c = country name

An example of an X.500 distinguished name:

```
"cn=someserver.company.ibm.com,o=IBM,ou=IBMGB,
   l=Winchester,s=Hants,c=GB"
```

-keypass *password*
The password used to protect the private key. Set this to the same value as the `-storepass` password, so that the CICS Transaction Gateway can 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, so that the CICS Transaction Gateway can establish a connection over SSL.

-hardwaretype *type*
The type of key pair that is being generated. Either CLEAR, PKDS or RETAINED. The default value is CLEAR.

-hardwareusage KEYMANAGEMENT
Sets the usage of the key pair being generated (SIGNATURE or KEYMANAGEMENT). The default value is KEYMANAGEMENT except for DSA keys, where it is SIGNATURE.

The following optional parameter is available when you are using the -genkey flag:

`-KeyLabel`
The label that will identify the private key on the hardware device. If this is not present a randomly-generated string is used.

The following optional parameter is available if you use -delete to delete a key:

`-hardwarekey`
Deletes the key pair from the hardware storage as well as the keystore. The default is that it is deleted only from the keystore.

The default keystore name when using hwkeytool is .HWkeystore in the user's home directory. Use the -keystore parameter to change this.

Every keystore file created by hwkeytool needs to have these items in the keystore:

- The personal certificate
- The Certificate Authority certificate used to sign it

If the personal certificate is self-signed, (created with the -selfcert parameter), first export the certificate and then import it into the same keystore file under a different alias. If you are warned when importing the certificate back into the keystore that it already exists in the keystore, type `Y` to confirm that you want to import it.

Using hardware cryptography

Cryptography within the existing JCE architecture gives Java 2 programmers security and performance advantages of hardware cryptography with minimal changes to existing Java applications.

To use hardware cryptographic function provided by the IBMJCECCA provider:

1. Edit the java.security file in the ${java-home}/lib/security directory so that it contains the following lines:
   ```
   security.provider.1=com.ibm.crypto.hdwrCCA.provider.IBMJCECCA
   security.provider.2=com.ibm.crypto.provider.IBMJCE
   ```
2. Copy the unrestricted policy files from the ${java-home}/demo/jce/policy-files/unrestricted directory to the ${java-home}/lib/security directory.

If you intend to use the keytool command to create JKS files that do not use hardware encryption:

1. Edit the java.security file to remove the line that references JCE4758.
2. Create the keystores.
3. If you intend to use hardware cryptography as well, reinstate the line in the java.security file.

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 SSL connection for a resource adapter:
   a. Define **`serverName`** as `ssl://<server_name>:<port>`.
   b. Set the keyRingClass property to the location of the key ring file.
   c. 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 138.
2. To enable SSL on each IPIC connection, set the **`ssl`** parameter in the “IPICSERVER section of the configuration file” on page 161 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.

Note: SP800-131A compliance is supplied in APAR PM98779 , PTF UK98510.

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`.

For strict support with .NET clients, the `SslGatewayConnection` 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 JVM must be updated with the Unrestricted JCE policy files placed in the 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 TXSeries versions which also support SP800-131a strict mode. This includes support for requests from 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

Configuring for client certificate mapping

CICS Transaction Gateway supports user ID and password authentication with RACF, in local and remote modes. Mapping of a registered X.509 certificate to a RACF user ID is supported in remote mode. RACF user ID and password authentication is used exclusively for EXCI connections to CICS; authentication of IPIC connections to CICS is performed by the CICS server.

For more information see, User authentication using SSL client certificates.

Preliminary checks

You use the `extattr +p` command to mark HFS files as program controlled. To use this command, you must be the owner of the files, or a superuser. The user ID that installed CICS Transaction Gateway normally owns the files. You also need READ access to the BPX.FILEATTR.PROGCTL FACILITY class. See *z/OS UNIX System*

Services Planning for more information. Your user ID must have the RACF SPECIAL attribute to perform the actions in step 3. Follow these steps to check that you have the necessary authority:

1. Log on to TSO.
2. Run ISPF.
3. Choose option 6 (Command).
4. Issue the following command:
   ```
   SR CLASS(FACILITY)
   ```
   Check that these entries are in the list:
   ```
   BPX.SERVER
   BPX.FILEATTR.PROGCTL
   ```
5. Issue the following command:
   ```
   SR CLASS(SURROGAT)
   ```
   Check that this entry is in the list:
   ```
   *.DFHEXCI
   ```

Configuring the system

1. Mark the load modules used by CICS Transaction Gateway as program-controlled. The HFS files which require the extended attribute +p are set correctly by SMP/E, however the SCTGLOAD and SDFHEXCI libraries must be set manually. All Java program files used by CICS Transaction Gateway during operation must also be program controlled to successfully run with authentication active. Use the "ls -E" command from an OMVS or Telnet screen to verify the CICS Transaction Gateway HFS files.
2. If necessary, activate program control by issuing these commands:
   ```
   SETROPTS CLASSACT(PROGRAM)
   RDEFINE PROGRAM * UACC(READ)
   SETROPTS WHEN(PROGRAM)
   ```
3. Mark the CICS SDFHEXCI library, which provides the EXCI for CICS Transaction Gateway, as program controlled. For example, if the library was installed as CICSTS51.CICS.SDFHEXCI, use the following RACF command:
   ```
   RALTER PROGRAM * ADDMEM('CICSTS51.CICS.SDFHEXCI'//NOPADCHK)
   SETROPTS WHEN(PROGRAM)REFRESH
   ```
4. Mark the CICS Transaction Gateway SCTGLOAD library, which provides the CTGBATCH program for CICS Transaction Gateway as program-controlled. For example, if this library was installed as CICSTG.CTG900.SCTGLOAD, use the following RACF command:
   ```
   RALTER PROGRAM * ADDMEM('CICSTG.CTG900.SCTGLOAD'//NOPADCHK)
   SETROPTS WHEN(PROGRAM)REFRESH
   ```
5. Mark the Language Environment® runtime library SCEERUN2 as program controlled. For example, if this library was installed as CEE.SCEERUN2, use the following RACF command:
   ```
   RALTER PROGRAM * ADDMEM('CEE.SCEERUN2'//NOPADCHK)
   SETROPTS WHEN(PROGRAM)REFRESH
   ```
6. Mark the CICS SDFHLINK library, which contains DFHRXSVC and DFHXCSVC, as program controlled.
7. Give the user ID under which CICS Transaction Gateway runs READ access to the BPX.SERVER FACILITY profile. For more information, see the `__passwd()` section in *z/OS V2R10.0 C/C++ Run-Time Library Reference*, (SC28-1663-08).

8. Ensure that the user ID that starts CICS Transaction Gateway has READ access to BPX.STOR.SWAP.
9. Give the user ID under which CICS Transaction Gateway runs READ access to the RACF profile that protects the TCPIP.STANDARD.TCPXLBIN data set. This contains tables for translating from ASCII to EBCDIC and from EBCDIC to ASCII.
10. Use one of the following options to configure CICS Transaction Gateway:

    Editing a STDENV file
    Ensure that this entry is in the file:

    ```
    AUTH_USERID_PASSWORD=YES
    ```

    Editing ctgenvvar
    Ensure that this entry is in the file:

    ```
    export AUTH_USERID_PASSWORD=Yes
    ```
11. If you are using CTGBATCH to start CICS Transaction Gateway, ensure that _BPX_SHAREAS=YES is set in the STDENV DD statement, regardless of whether a ctgenvvar script is also being used. If starting CICS Transaction Gateway from USS, set _BPX_SHAREAS=NO in the ctgenvvar script, to force the use of a clean address space.

Related information:

"Security error due to surrogate checking problem" on page 283
An ECI_ERR_SECURITY_ERROR -27 can occur if a user ID is not authorized as a surrogate for the user ID specified on the ECI request.

Configuring identity propagation

Identity propagation configuration tasks are required on RACF, CICS Transaction Server and WebSphere Application Server. Identity propagation must also be activated in CICS Transaction Gateway.

Configuring identity propagation on RACF

The steps required to configure RACF for identity propagation.

RACF must contain mappings of distinguished names to RACF user IDs. The distinguished names defined in the mappings must have the same format as they have in the user registry.

For more information about configuring IPIC connections and RACF, see the CICS Transaction Server Information Center.

A command RACMAP is available for creating, deleting, and listing a distributed identity filter. If changes are required, you can delete the filter, and define a new one. The RACMAP command has the following functions:

MAP	creates a distributed identity filter
DELMAP	deletes a distributed identity filter
LISTMAP	lists information about a distributed identity filter

Examples:

```
RACMAP ID(GUSKI) MAP
   USERDIDFILTER(NAME('UID=RICH,OU=Web Sales,O=Rich Radio Ham,L=Internet'))
   REGISTRY(NAME('us.richradioham.com'))
   WITHLABEL('Rich''s name filter')
```

```
RACMAP ID(SMITH) MAP
  USERDIDFILTER(NAME('uid=JIM,ou=Web Sales,dc=CTGSales, o=HEADOFFICECTG')) -
  REGISTRY(NAME('uk.websales.com'))
```

For more information about the RACMAP command, see the*z/OS Security Server RACF Command Language Reference*.

Note: It is not possible to modify a distributed identity filter.

Configuring identity propagation on CICS

The steps required to configure identity propagation on CICS Transaction Server.

CICS Transaction Server requires the following:
- The z/OS identity propagation function provided in z/OS, Version 1.11 or later
- CICS Transaction Server for z/OS Version 4.1 or later with the APAR fixes described in "Configurations that support identity propagation" on page 51. To download these fixes, go to Fix list for CICS Transaction Server for z/OS V4.1
- An IPIC connection with USERAUTH set to IDENTIFY

If the CICS Transaction Gateway making the request and the CICS server are not in the same sysplex (for example when a resource adapter using a local Gateway issues requests directly to CICS), an SSL connection is required to allow the use of USERAUTH=IDENTIFY. For more information, see the "User security" section of "IPIC connection security" on page 43.

Configuring identity propagation on WebSphere Application Server

Configuration is required on WebSphere Application Server to enable identity propagation.

Setting up the identity propagation login module

WebSphere Application Server must be configured to specify a user registry to enable user ID and password verification for applications. Any registry supported by WebSphere Application Server is supported by CICS Transaction Gateway. Examples of the registries supported by WebSphere Application Server are:
- IBM Tivoli Directory Server (ITDS)
- Microsoft Active Directory
- SunOS Directory
- Novel Directory Service

For more information about supported registries, see the WebSphere Application Server Information Center.

All JEE applications that call the CICS Transaction Gateway ECI resource adapter must be configured for container-managed security.

CICS Transaction Gateway includes a JAAS (Java Authentication and Authorization Service) login module in the ECI resource adapter RAR (cicseci.rar). You must install the login module into WebSphere Application Server to enable identity propagation. Install the login module by creating a new JAAS Application Login alias that refers to the fully-qualified name of the login module:

```
com.ibm.ctg.security.idprop.LoginModule
```

One of the following must be configured to use the CICS Transaction Gateway identity propagation login module:

- The JEE application must be configured to use a custom login configuration that refers to the CICS Transaction Gateway identity propagation login module. This is accessed via the connection factory resource references on the application's configuration panel.
- The connection factory that is used by the application must have a mapping configuration alias that refers to the CICS Transaction Gateway identity propagation login module. This is accessed by the connection factory's configuration panel.

For more information about configuring WebSphere Application Server, see the WebSphere Application Server Information Center.

Specifying the authentication information to propagate

If identity propagation has been configured and activated, the identity information that can be propagated with a request can be either the identity of the user who invoked the application, or the identity under which the application programmer has configured the application to run.

- The identity of the user who invoked the application is known as the “caller” or “received” identity.
- The identity under which the application programmer has configured the application to run is known as the “run as” or “invocation” identity.

To specify the identity to propagate to CICS, you set the `propIdentity` custom property on the CICS Transaction Gateway identity propagation login module. You do this from the WebSphere Application Server admin console by setting one of the following name-value pairs:

```
propIdentity=Caller
```

or

```
propIdentity=RunAs
```

For example, if you want the “run as” identity to be propagated to CICS, do this:

1. From the WebSphere administrative console; 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 WebSphere Application Server. For more information, see "Deploying the CICS resource adapter" on page 169.
2. Configure an IPIC server definition from CICS Transaction Gateway into a CICS server on the same sysplex. Alternatively you can configure an IPIC connection from a local Gateway directly into CICS, using SSL.
3. Install the CICS Transaction Gateway identity propagation login module in WebSphere Application Server. For more information, see "Configuring identity propagation on WebSphere Application Server" on page 146.
4. Configure the Java client application resource references, or the connection factories used by the applications, to use the CICS Transaction Gateway identity propagation login module. When applications have been enabled to use the module, identity propagation is active. For more information about configuring WebSphere Application Server, see the WebSphere Application Server Information Center.

Configuring high availability

High availability is supported by the default server, policy-based dynamic server selection (DSS), the CICS request exit, and the logical CICS server definition (deprecated).

The logical CICS server definition has been superseded by the policy-based DSS definition. For information about migration see "Logical CICS server definitions" on page 29.

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 by the Gateway daemon for application requests in which no CICS server name is specified. The name can be a logical CICS server name.

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

Default value
There is no default value for this parameter.

Configuring a dynamic server selection policy

Policy-based dynamic server selection provides a set of configurable rules that determine the destination CICS servers for ECI and ESI requests.

To determine which policy the Gateway daemon uses at runtime, you set the DSSPOLICY parameter in the configuration file. For more information see the "GATEWAY section of the configuration file" on page 158.

Setting the active DSS policy

The **dsspolicy** parameter specifies the name of the DSS policy to use for dynamic server selection.

dsspolicy=<name>

Description
Set the value to name a DSSPOLICY section in the configuration file. To switch policies, you must shut down the Gateway daemon, edit the configuration file and restart the Gateway daemon.

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

Default value
The default is that there is no DSS policy.

Configuring a DSS policy

To configure a DSS policy, you add a DSSPOLICY section to the Gateway daemon configuration file. The DSSPOLICY section defines the policy name, and the mappings between logical servers and DSS groups (groups of CICS servers).

Each DSS policy must have a name that is unique within a Gateway daemon configuration. The active policy is specified by the **dsspolicy** parameter in the GATEWAY section.

The mappings themselves are defined in the MAPPINGS subsection of the DSSPOLICY section. For more information, see "Mappings."

Here is an example of a DSSPOLICY section:

```
SECTION DSSPOLICY = POLICY1
   SUBSECTION MAPPINGS
      CICSX=GROUP1
      <NONE>=GROUP2
   ENDSUBSECTION
ENDSECTION
```

Mappings:

The mappings in the DSS policy associate logical CICS server names with DSS groups (groups of CICS servers).

The MAPPINGS subsection must contain at least one mapping.

`<NONE>` mapping

If you want a logical CICS server name to be applied to all requests that do not specify a CICS server name, you must use the value **`<NONE>`** as the logical CICS server name. If you specify both a **`<NONE>`** mapping and a default server in the configuration, the **`<NONE>`** mapping takes precedence.

In the example, requests that specify the logical CICS server name CICSX are mapped to the DSS group GROUP1. Requests that do not specify a server name are mapped to GROUP2 because of the **`<NONE>`** mapping. For more information about DSS groups, see "Configuring a DSS group" on page 150.

```
SECTION DSSPOLICY = POLICY1
   SUBSECTION MAPPINGS
      <NONE>=GROUP2
      CICSX=GROUP1
   ENDSUBSECTION
ENDSECTION
```

`<ANY>` mapping

If you want a logical CICS server name to be applied to all requests that do not match any of the mappings in the policy, you must specify an **`<ANY>`** mapping. If you configure an **`<ANY>`** mapping, it is not possible to send requests directly to a CICS server.

In the example, requests that specify the logical CICS server name CICSX or CICSY are mapped to the DSS group GROUP1. Requests that do not specify a server name are mapped to GROUP2 because of the **`<NONE>`** mapping. All other requests are sent to GROUP3.

```
SECTION DSSPOLICY = POLICY1
   SUBSECTION MAPPINGS
      <ANY>=GROUP3
      CICSX=GROUP1
      CICSY=GROUP1
      <NONE>=GROUP2
   ENDSUBSECTION
ENDSECTION
```

Configuring a DSS group

To configure a DSS group, you add a DSSGROUP section to the Gateway daemon configuration file. The DSSGROUP section defines the group name, the CICS servers that belong to the group, and the algorithm that CICS Transaction Gateway uses to select a CICS server within the group.

Each DSSGROUP section within a Gateway daemon configuration must have a unique name, otherwise the configuration is non-valid and the Gateway daemon fails to start.

Here is an example of a DSSGROUP section:

```
SECTION DSSGROUP = GROUP1
   Servers=CICSA,CICSB,CICSC,CICSD
   Algorithm=ROUNDROBIN
ENDSECTION
```

CICS servers in DSS group:

The **`Servers`** parameter defines the names or APPLIDs of the CICS servers that belong to the DSS group.

`Servers=<name1[,name2...]>`

Description

Specify one or more SECTION IPICSERVER server names or APPLIDs of EXCI connected servers in a comma-separated string. If XA support is enabled, SECTION IPICSERVER server names cannot be mixed with APPLIDs of EXCI connected servers within the same DSS group.

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

Default value

There is no default value.

Dynamic server selection algorithm:

The **`Algorithm`** parameter specifies the algorithm that CICS Transaction Gateway uses to select a CICS server within a DSS group.

`Algorithm=<roundrobin|failover>`

Description
Set the value to **`roundrobin`** to specify that transactions are distributed evenly between the CICS servers in the DSS group. If the selected CICS server is not available, then the other CICS servers in the list are tried in turn until an available server is found or all of the servers have been tried. If you set the value to **`failover`**, the first CICS server in the DSS group is selected by default. If the first server is not available, then each server in the list is tried in turn until an available server is found or there are no more servers in the list.

This parameter is in the "DSSGROUP section of the configuration file" on page 163.

Default value
There is no default value.

Configuring a 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 "CICS request exit" on page 78.

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

Default value
There is no default value.

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=`** option on the **`ctgstart`** command to override the value of **`requestexits`**. For more information, see Command reference.

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

Default value
There is no default value for this parameter.

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 6. 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 170
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

Edit the GATEWAY section of the configuration file to configure the Gateway daemon monitoring resources.

Statistics API protocol settings

To configure the statistics API protocol settings, edit the statistics API protocol parameters in the GATEWAY section of the configuration file.

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 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.

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.

You can use the **`ctgstart`** command with the *-statsport* option to override the value of the **`port`** parameter. For more information, see “Options on the ctgstart command” on page 254.

This parameter is in the “Statistics API protocol parameters” on page 161 subsection of the GATEWAY section of the configuration file.

Default value
If this parameter is not specified, the default value is 2980.

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 “Statistics API protocol parameters” on page 161 subsection of the GATEWAY section of the configuration file.

Default value
If this parameter is not specified, the default value is 2000 milliseconds.

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 161 subsection of the GATEWAY section of the configuration file.

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

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.

Default value
If this parameter is not specified, the default value is 030000.

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.

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

Enable statistic recording to SMF

The **statsrecording** parameter determines whether statistics are written to the System Management Facility (SMF) on z/OS.

statsrecording=<on|off>

Description

Set the value to **on** to enable recording of statistics. To write to SMF, the user ID that the CICS Gateway daemon runs, must have READ access to the BPX.SMF facility. You configure this permission during the installation or upgrade processes. For further details, see the RACF and UNIX System Services (USS) documentation.

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

Default value

The default for this parameter is off.

Health reporting

The Gateway daemon can monitor certain error codes to determine the health of communications with CICS.

Health reporting can be enabled for use in TCP/IP load balancing topologies, with CICS Transaction Gateway running in remote mode. The interval used for this monitoring is specified by the health interval. Health reporting requires the IWM4HLTH macro.

The load balancer can use health information to set priorities when creating new incoming IP socket connections to Gateway daemons in the load balancing group. To enable TCP/IP port sharing to use the health information for load balancing decisions, set SHAREPORTWLM on the PORT definition. To enable Sysplex Distributor to use the health information for load balancing decisions, set SERVERWLM on the VIPADISTRIBUTE statement.

For information about how health is calculated, see “Health monitoring” on page 74.

Related concepts:

“Health monitoring” on page 74
A TCP/IP load balancer that is allocating a connection to the Gateway daemon detects whether or not a CICS server is available. The Gateway daemon reports the health of its CICS server connections to the TCP/IP load balancer.

Related tasks:

“Determining health status” on page 260
The current health status is available in the GD_CHEALTH statistic. This information describes how to find the current health status.

Related information:

“Resetting health status” on page 260
This information describes how to reset the health status to 100.

Enable health reporting:

The **healthreporting** parameter is used to specify that the Gateway daemon reports the health of communications with CICS to a TCP/IP load balancer.

healthreporting=<on>

Description

Set the value to **on** to enable reporting of the health of communications with CICS to a TCP/IP load balancer. If you do not set `healthreporting` parameter to **on**, statistics relating to the current health of communications with CICS are still collected by the Gateway daemon, but they are not reported to the TCP/IP load balancer.

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

Default value

By default, this parameter is not included in the configuration file.

Health interval:

The `healthinterval` parameter specifies the amount of time, in seconds, that the Gateway daemon monitors error codes that are associated with the health of communications with CICS.

`healthinterval=<number>`

Description

Set the value in the range 1 - 9,999 to specify the monitoring period in seconds. After this monitoring period, the connection health is calculated and reported to the WLM.

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

Default value

The default value is 60 seconds.

Configuring bidirectional data support

Use the -Dctg.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 text support use the **`ctgstart`** command to start CICS Transaction Gateway with the following option:

```
ctgstart -j-Dctg.bidi.target.layout=RTL
```

If you start CICS Transaction Gateway without the system property option described above, by default, bidi data support is off. However, you can explicitly turn off bidi support by specifying -Dctg.bidi.target.layout=OFF.

Configuring trace settings

Edit the trace attributes in the GATEWAY section of the configuration file.

Configuration parameter reference

The way in which you configure CICS Transaction Gateway depends on how the Gateway daemon is to be started.

Sample configuration files are provided in the <install_path>/samples/ configuration directory:

Sample version (<sample version>)	Used by
ctgsamp.ini	The Gateway daemon.
CTGENV Supplied in MVS data set format SCTGSAMP(CTGENV)	The Gateway daemon when started using JCL and CTGBATCH, or applications started by CTGBATCH and using the CICS Transaction Gateway in local mode.
ctgenvvarsamp	The Gateway daemon when started from the USS command line interface, or USS applications using the CICS Transaction Gateway in local mode.

Use the *CICSCLI* and *CTGENVVAR* environment variables to specify the location of these files. The default location for the configuration files is <install_path>/bin/.

Note: Some lines of the configuration file are longer than 72 characters; take care when editing them.

In the following sections relevant settings for each file are shown, together with their corresponding definition in *Configuring a remote mode topology*.

Related information:

"STDENV file" on page 98
The STDENV file can be used if the Gateway daemon is to be started in batch mode using CTGBATCH to define the required environment variables.

The configuration file

An HFS file or MVS dataset can be used to initialize the CICS Transaction Gateway.

The default configuration used by CICS Transaction Gateway is the HFS file `<install_path>/bin/ctg.ini` You can modify the directory and filename if required. For more information see the *CICSCLI* environment variable in "Environment variables: remote mode" on page 102.

To denote comments in the configuration file use the hash (#) character. The hash (#) character must be positioned at the start of the line, or must have a space or tab character before it.

To split definitions over multiple lines, use backslash (\) as a continuation character. The continuation character should be the placed immediately after the last non-blank character on the line.

Creating a configuration

A sample HFS configuration file and a MVS dataset configuration are supplied with the product. These sample configurations can be copied and modified as required. The sample HFS configuration file is located in `<install_path>/samples/configuration/ctgsamp.ini`. The sample MVS dataset member is CTGCONF in the CTGSAMP dataset.

Sections within the configuration

The configuration file must contain at least the GATEWAY section with a TCP protocol handler definition, otherwise CICS Transaction Gateway will not start. A section begins with a SECTION element and must be terminated with an ENDSECTION element. Each SECTION element must be positioned as the first entry on a given line.

Configuration parameters

For information about configuration parameters and supported values see the parameter descriptions in "Configuration parameter reference" on page 156.

Default values are automatically applied to parameters that are not explicitly defined.

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 7. SECTION PRODUCT

Entry in the configuration file	Description
applid	"Gateway APPLID" on page 107
applidqualifier	"Gateway APPLID qualifier" on page 107
defaultserver	"Default server" on page 148
esmkeyring	"ESM key ring" on page 139
hwcrypt	"Use hardware cryptography" on page 139
keyring	"Key ring file" on page 138
keyringpw	"Key ring password" on page 138
keyringpwscrambled	"Key ring password encryption" on page 138

GATEWAY section of the configuration file

This table provides the names and descriptions for all parameters that can be set in the GATEWAY section of the configuration file.

There must be no more than one GATEWAY section in the configuration file.

Table 8. SECTION GATEWAY

Entry in the configuration file	Description
cicslogging	"Log CICS messages" on page 87
cicsrequestexit	"Configuring a CICS request exit" on page 151
closetimeout	"Timeout for in-progress requests to complete" on page 86
connectionlogging	"Log Client connections and disconnections" on page 87
dnsnames	"Display TCP/IP hostnames" on page 87
dsspolicy	"Setting the active DSS policy" on page 149

Table 8. SECTION GATEWAY (continued)

Entry in the configuration file	Description
dumpoffset	"Data byte offset in trace data" on page 95
healthinterval	"Health interval" on page 156
healthreporting	"Enable health reporting" on page 155
initconnect	"Initial number of connection manager threads" on page 84
initworker	"Initial number of worker threads" on page 84
maxconnect	"Maximum number of connection manager threads" on page 84
maxworker	"Maximum number of worker threads" on page 85
noinput	"Enable reading input from SDSF" on page 86
requestexits	"Configuring request monitoring exits for the Gateway daemon" on page 151
stack	"Exception stack tracing" on page 96
start	"Gateway start type" on page 131
stateod	"Statistics end of day time" on page 154
statint	"Statistics interval" on page 154
statsrecording	"Enable statistic recording to SMF" on page 155
tfile	"Gateway trace file" on page 95
tfilesize	"Gateway trace file wrap size (KB)" on page 95
trace	"Enable Gateway daemon trace on startup" on page 96
truncationsize	"Maximum size of trace data blocks" on page 96
workertimeout	"Worker thread availability timeout" on page 86
xasupport	"XA transaction support activate (**`xasupport`**)" on page 131.

TCP protocol parameters

To enable the TCP protocol, add the name of the TCP protocol handler to the GATEWAY section of the 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=<number>
;dropworking;\idletimeout=<number>;pingfrequency=<number>;port=<number>;
requiresecurity;\    solinger=<number>;
```

Entries for each protocol must be in the form shown:

- Two lines are allowed for each protocol. 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 TCP settings panel:

Table 9. TCP protocol

Entry in the ctg.ini file	Description
bind	“Bind address” on page 88
connecttimeout	“Connection timeout” on page 89
dropworking	“Drop working connections” on page 90
idletimeout	“Idle timeout” on page 89
pingfrequency	“Ping frequency interval” on page 89
port	“Port” on page 88
requiresecurity	“Require Java Clients to use security classes” on page 90
solinger	“SO_LINGER setting” on page 90

SSL protocol parameters

To enable the SSL protocol, add the name of the SSL protocol handler to the GATEWAY section of the 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>;keyring=<file>;\
keyringpw=<password>;keyringpwscrambled=<on|off>;\
pingfrequency=<number>;port=<number>;requiresecurity;solinger=<number>;\
ciphersuites=<name>;
```

Note that you do not need to specify the parameters that are not required for your configuration.

Entries for each protocol must be in the form shown:
- Two lines are allowed for each protocol. 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 10. SSL protocol

Entry in the configuration file	Description
bind	
ciphersuites	“Use only these ciphers” on page 94
clientauth	“Use client authentication” on page 94
connecttimeout	“Connection timeout” on page 89
dropworking	“Drop working connections” on page 90
idletimeout	“Idle timeout” on page 89
pingfrequency	“Ping frequency interval” on page 89
port	“Port” on page 91
requiresecurity	“Require Java Clients to use security classes” on page 90

Table 10. SSL protocol (continued)

Entry in the configuration file	Description
solinger	"SO_LINGER setting" on page 90

Note: For a description of the parameters **`esmkeyring`**, **`hwcrypt`**, **`keyring`**, **`keyringpw`**, and **`keyringpwscrambled`**, see PRODUCT section of the configuration file.

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. 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 11. Statistics API protocol parameters

Entry in the ctg.ini file	Description
bind	"Bind address" on page 153
connecttimeout	Connection timeout
port	"Port" on page 153
maxconn	"Maximum number of connections" on page 154

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 12. SECTION IPICSERVER

Entry in the configuration file	Description
SECTION IPICSERVER	"Server name" on page 114
cicsapplid	"Target CICS APPLID" on page 116
cicsapplidqualifier	"Target CICS APPLID qualifier" on page 116
connecttimeout	"Connection timeout" on page 117

Table 12. SECTION IPICSERVER (continued)

Entry in the configuration file	Description
description	"Description" on page 30
ecitimeout	"ECI timeout" on page 118
hostname	"Host name or IP address" on page 115
port	"Port" on page 115
sendsessions	"IPIC send sessions" on page 115
srvidletimeout	"Server idle timeout" on page 117
srvretryinterval	"Server retry interval" on page 117
tcpkeepalive	"Send TCP/IP KeepAlive packets" on page 118
ssl	"Use SSL" on page 118
ciphersuites	"Use only these ciphers" on page 118

LOGICALSERVER section of the configuration file

The LOGICALSERVER section of the configuration file defines the mapping between a logical CICS server name and the name of an actual CICS server.

LOGICALSERVER section definitions are deprecated and superseded by policy-based dynamic server selection definitions. For more information on migrating definitions see Logical CICS(r) server definitions.

Table 13. SECTION LOGICALSERVER

Entry in the configuration file	Description
SECTION LOGICALSERVER=<Server name>	"Server name" on page 30
description	Description
server	"CICS server name" on page 30

The following template shows the configuration file definition for a logical CICS server:

```
SECTION LOGICALSERVER = CICS1
  DESCRIPTION=ServerA Alias
  SERVER=CICSA
ENDSECTION
```

DSSPOLICY section of the configuration file

The DSSPOLICY section of the configuration file defines the dynamic server selection (DSS) mappings between logical CICS server names and actual CICS servers.

Table 14. SECTION DSSPOLICY

Entry in the configuration file	Description
SECTION DSSPOLICY=<policy_name>	Configuring a DSS policy (DSSPOLICY)
SUBSECTION MAPPINGS	Mappings

Here is an example of a DSSPOLICY entry in the configuration file:

```
SECTION DSSPOLICY = POLICY1
   SUBSECTION MAPPINGS
      CICSX=GROUP1
      <NONE>=GROUP2
   ENDSUBSECTION
ENDSECTION
```

DSSGROUP section of the configuration file

The DSSGROUP section of the configuration file defines the mapping between a logical CICS server name and one or more actual CICS servers, and the algorithm used for selecting an actual CICS server.

Table 15. SECTION DSSGROUP

Entry in the configuration file	Description
SECTION DSSGROUP=<group name>	Configuring a DSS group (DSSGROUP)
servers	CICS servers in DSS group (**`servers`**)
algorithm	Dynamic server selection algorithm (**`algorithm`**)

Here is an example of a DSSPOLICY entry in the configuration file:

```
SECTION DSSGROUP = GROUP1
   Servers=CICSA,CICSB,CICSC,CICSD
   Algorithm=ROUNDROBIN
ENDSECTION
```

Summary of environment variables

Environment variables control how CICS Transaction Gateway functions.

The table provides further details about the fields in the configuration file, see Configuring.

Table 16. Environment variables

Environment variable	Description
AUTH_USERID_PASSWORD	Specifies whether the user ID and password is authenticated with RACF.
BPX_SHAREAS	Specifies whether or not all processes involved in starting the Gateway daemon run in a single address space.
BPXK_SETIBMOPT_TRANSPORT	Specifies the job name of the TCP/IP stack to be used by the Gateway daemon.
CICSCLI	Specifies a runtime path and file name for the configuration file, ctg.ini, used in remote mode.
COLUMNS	Specifies the maximum line-length for messages output to the console when the CICS Transaction Gateway is started.
CTG_EXCI_INIT	Specifies whether or not EXCI is loaded. If this variable is set to YES, EXCI is loaded.
CTG_JNI_TRACE	Sets the name of the JNI trace file.
CTG_JNI_TRACE_ON	Specifies whether or not JNI trace is enabled.
CTG_MIXEDCASE_PW	Specifies whether or not mixed-case passwords are authenticated.

Table 16. Environment variables (continued)

Environment variable	Description
CTG_PIPE_REUSE	Specifies whether all allocated EXCI pipes are reused by CICS Transaction Gateway, or only a maximum of one is reused per worker thread.
CTG_RRMNAME	Specifies the name of the resource manager managing this instance of the CICS Transaction Gateway.
CTG_SWAPPABLE	Specifies whether the address space where CICS Transaction Gateway runs is swappable or nonswappable.
CTG_WIDTH	Specifies the maximum width of the STDENV script output.
CTG_XA_MAX_TRAN	Set this environment variable to limit the maximum number of concurrent XA transactions that can use the EXCI protocol in a Gateway daemon.
CTGENVVAR	Specifies the fully qualified location of a "ctgenvvar" script to be invoked by ctgstart.
CTGSTART_OPTS	Specifies options that are too long for inclusion in the JCL step.
DFHJVPIPE	Specifies the name of the specific pipe that CICS Transaction Gateway uses for EXCI calls.
DFHJVSYSTEM_*nn*	Specifies the name and description of an EXCI connected CICS server to be returned in response to a request for the CICS_EciListSystems function.
PATH	Specifies the path in HFS containing the runtime resources necessary to run ctgstart.
STEPLIB	Specifies the library containing the default EXCI options and the EXCI load modules.
TMPDIR	Specifies a temporary directory other than /tmp.
TZ	Specify the local time zone and daylight saving time.

Testing your configuration

Run a test to check that CICS Transaction Gateway has been configured correctly.

- Run the local mode or remote mode test batch jobs to check the configuration and connectivity from the CICS Transaction Gateway components to CICS.
- Run the JCA resource adapter installation verification test to verify whether the CICS Transaction Gateway ECI resource adapter can be used with your JEE 1.4 or JEE 5, or later, application server.
- Run the sample programs supplied with CICS Transaction Gateway.

Using the sample batch jobs to check your configuration

To test your configuration, use the sample JCL stored in the SCTGSAMP library members CTGTESTL and CTGTESTR.

Use CTGTESTL to test a local mode topology and CTGTESTR to test a remote mode topology. Modify these sample jobs to add details of the data set high-level qualifiers and HFS paths for your installation.

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 an application server as follows:

- The CICS Transaction Gateway V9.0 ECI resource adapter (cicseci.rar) with a JEE 6 certified application server
- Resource adapters supplied in Supportpac CC03 with a JEE 1.4 certified application server or JEE 5 certified application server

The IVT runs as a servlet within a JEE application server and calls program EC01 on the CICS server. The IVT sends two ECI requests to CICS:

1. A non-transactional request, which is not coordinated by the transaction manager.
2. A transactional request, which uses the global transaction support provided by the application server.

IBM has successfully tested the ECI resource 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.

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 1.4 or JEE 5 (or later) compatibility test suite. For more information see:

`http://java.sun.com/javaee/overview/compatibility.jsp`

Prerequisites

Complete the following tasks:

- Compile and install the EC01 sample CICS COBOL program on the CICS server.
- Ensure that the CICS Transaction Gateway resource adapter archive RAR file (cicseci.rar) is available.
- Ensure that the JCA IVT enterprise archive (EAR) file is available and has the filename ECIIVT.ear.
- Configure a CICS server connection for CICS Transaction Gateway.

The source for EC01 is shipped with CICS Transaction Gateway as a COBOL file in <install_path>/samples/server/ec01.ccp. Both the RAR files and the EAR files are shipped with the CICS Transaction Gateway in the <install_path>/deployable directory.

To ensure you have correctly configured the CICS Transaction Gateway, follow the instructions in the *CICS Transaction Gateway for z/OS: Application Programming Guide* to run the EciB1 sample program. The sample program can be found in <install_path>/samples.

Deploying and configuring the JCA IVT

To 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 EXCI, set the server name to the CICS APPLID.
   - For all other configurations, set the server name to the server defined in the configuration file (ctg.ini).
4. Install the application ECIIVT.ear with a target resource JNDI name of ECIIVTBean1. The ECIIVT.ear is located within the <install_path>/deployable directory.

Running the JCA IVT

To run the JCA IVT.

1. Use a web browser to display the first IVT Web page index.jsp. 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 for details, including compilation instructions and information on compiler considerations.

Chapter 9. Deploying applications

The method you must use to deploy your application depends on the CICS TG API used by the application.

Configuring remote Client application environments

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

Java Client applications

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

To compile and run Java applications on a client machine, add the full path of `ctgclient.jar` to the *CLASSPATH* environment variable. This archive is in the `<install_path>/classes` 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.

C Client applications

The files required for compiling and running C applications on a client machine are in the `ctgredist.tar.gz` package on UNIX and Linux or the `ctgredist.zip` package on Windows, these are found in the `<install_path>/deployable` directory. Copy the package to the client machine before extracting.

To compile ECI version 2 C applications on a Windows client machine you must include the files `ctgclient_eci.h`, `ctgclient.h`, and `ctgclient.lib` in your C build environment. To run the applications `ctgclient.dll` is required in the path.

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

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

Deploying the CICS resource adapter

The resource adapter is provided as a standard module, ready for deployment into a Java Platform, Enterprise Edition (JEE) application server. The resource adapter 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 adapter which is located in the `<install_path>/deployable` directory:

- ECI resource adapter (cicseci.rar)

The resource adapters can be deployed in 31-bit and 64-bit runtime environments. For more information on supported environments, see “JEE application servers” on page 10.

For information on how to deploy the 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 Programming Guide*.

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`.

  Local transactions are not supported when using WebSphere Application Server for z/OS with CICS Transaction Gateway for z/OS in local mode, as the resource adapter provides global transaction support with MVS RRS.
- 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 EXCI 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 EXCI and IPIC for CICS connections. In remote mode, this field is used to identify the client connection to the Gateway daemon.

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

tcp
ssl
local

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

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

serverName
The name of the CICS server to connect to for all interactions through this resource adapter. In remote mode, this name must be defined in the CICS Transaction Gateway configuration file. If this parameter is left blank, the default CICS server is used; For more information see “PRODUCT section of the configuration file” on page 158. 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 z/OS: Programming Guide*.

serverSecurity
The fully-qualified name of the ServerSecurity class to use in each interaction with CICS. This parameter is optional; if no value is given, no ServerSecurity class is used. 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 z/OS: Programming Guide*.

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

keyRingPassword
The password for the keystore defined in keyRingClass.

traceLevel
The level of trace to be output by the resource adapter. For more details on trace levels and tracing see “JEE tracing” on page 297.

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 z/OS: Programming Guide*.

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 z/OS, whichever is lower.

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 cicseci resource adapter supports 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.

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

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

Deploying the ECI resource adapter on WebSphere Application Server for z/OS

You must completely remove any resource adapter supplied with a previous version of the product before installing this version.

Local mode

If you are using local mode, the native path of the resource adapter must point to the `bin` directory of the CICS Transaction Gateway installation.

To create this path on WebSphere Application Server:

1. Navigate to **Resources > Resource Adapters > Resource Adapters**.
2. Click the specific resource adapter.
3. In the **Native path** box enter the full path to the CICS Transaction Gateway <install_path>/bin directory, for example:
   `/usr/lpp/cicstg/ctg900/bin`

You must also set the environment variables that are relevant to JCA in WebSphere Application Server. For a local mode topology, the CICS Transaction Gateway environment variables are specified using the WebSphere Administration Console:

- CTG_EXCI_INIT determines whether or not EXCI is loaded (default is YES if undefined). If EXCI is required, you must also set the STEPLIB environment variable. You can also set CTG_PIPE_REUSE and DFHJVPIPE:

  - STEPLIB identifies the library containing the default EXCI options and the EXCI load modules (mandatory).
  - CTG_PIPE_REUSE determines how allocated EXCI pipes are reused (optional but recommended).
  - DFHJVPIPE defines the name of the pipe that CICS Transaction Gateway uses for EXCI calls (optional but recommended).

For more information about what values to set for these environment variables see "Environment variables: local and remote mode" on page 100.

For more information about local mode see "Configuring a local mode topology" on page 81.

Deploying remote Java client 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>\classes directory:

Ensure that any JAR files that you copy are listed on the class path of the remote computer.

Deploying ECI V2 and ESI V2 to remote systems

Remote ECI V2 and ESI V2 applications are deployed as executable files. The API is not supported on z/OS but connectivity from a remote ECI V2 or ESI V2 application to CICS TG for z/OS is supported.

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 ctgredist package.

You can find a 64-bit version of this file in the `<platform>/lib64` directory of the ctgredist package.

On 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.

Deploying .NET applications to remote systems

Remote .NET 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 .NET API assembly: IBM.CTG.Client.dll

The CICS Transaction Gateway .NET API supports 32-bit and 64-bit operation. Support is provided by a single assembly (IBM.CTG.Client.dll) which is included in the ctgredist package in the directory `Windows/lib`.

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

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

Chapter 10. Scenarios

Follow the steps in these scenarios to learn how to perform tasks such as configuring connections to CICS, or configuring SSL security. As you work through each scenario you use real values provided in a reference table. When you have completed the configuration part of a scenario, you can then test the scenario by sending a simple ECI request to a CICS server.

Sample files

Sample files containing CICS Transaction Gateway configuration parameter values and environment variables are provided for the scenarios. The sample files are installed as part of the product package, and can be accessed from the scenarios through download links.

Each sample file contains values specific to that scenario; The sample files that contain configuration parameters have the common file name ctg.ini. The sample files that contain environment variables have unique filenames such as CTGS05NV. Environment variables are not required for all scenarios.

To open a sample file and view the contents:

1. Open the introductory topic for the scenario, for example "Configuring a secure autoinstalled IPIC connection (SC01)."
2. Scroll down to the sample file link (the link is located immediately below the table of values).
3. Double-click the link.

To download and save a sample file onto your local machine:

1. Right-click the sample file link and select **Save Target As..**.
2. Specify the location where you want to save the sample file.

The samples for MVS are shipped in the SCTGSAMP product library and use the following naming convention:

```
hlq.SCTGSAMP(CTGSnnXX)
```

Where *nn* is the number of the scenario that uses that sample and *XX* is a 2-character identifier.

The samples for HFS are installed in the following location:

```
<install_path>/samples/scenarios/scnn
```

Where `nn` is the scenario number, for example:

```
<install_path>/samples/scenarios/sc01
```

Configuring a secure autoinstalled IPIC connection (SC01)

You can configure secure autoinstalled IPIC connections using a template. Using a template allows you to change the default connection settings for IPIC autoinstalled connections. To implement an IPCONN template so that IPIC connections are autoinstalled with link security and user security, follow the step-by-step instructions in this scenario.

To configure secure autoinstalled IPIC connections, you must modify the CICS TS sample user-replaceable module (URM) to point to an IPCONN template.

This scenario uses CICS TG connecting to CICS TS V3.2 over IPIC in remote mode. It uses the default name `ctg.ini` for the configuration file.

Table 17. Values used in this scenario

Component	Parameter	Where set	Example value	Matching values
CICS TG	Server name	IPICSERVER section of ctg.ini	CICSA	
CICS TG	Hostname	IPICSERVER section of ctg.ini	cicssrv2.company.com	
CICS TG	Port **1**	IPICSERVER section of ctg.ini	50889	This value must be the same as **3**
CICS TS	IPCONN template	In DFHISCIP (autoinstall user program)	SECTEMPL	
CICS TS	TCPIPService **2**	TCPIPService definition	SRV50889	This value must be the same as **4**
CICS TS	Portnumber **3**	TCPIPService definition	50889	This value must be the same as **1**
CICS TS	TCPIPService **4**	IPCONN definition	SRV50889	This value must be the same as **2**
RACF	User ID for link security	IPCONN definition in CICS TS	LINKUSER	
RACF	User ID for user security	Client application	USERID	
RACF	Password for user security	Client application	PASSWORD	

Prerequisites

You must satisfy these system requirements.

Here are the system requirements for CICS TS for z/OS:

- The server must be CICS V3.2 or later because IPIC is not available in earlier releases of CICS.
- TCP/IP services must be active in the CICS server.
  - To activate these services, set the 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.

- The TCP/IP network must extend between LPARs if CICS TG for z/OS and the CICS server exist on different LPARs.
- You must set the SEC system initialization parameter to YES to enable security.
- You must have valid RACF user IDs and passwords.

Here are the system requirements for CICS TG:
- CICS TG must be installed.

To test that the scenario works successfully you can use either the supplied samples, or your own applications. If you use the supplied samples, this scenario requires:
- The sample CICS TG server program EC01 must be compiled, defined, and installed on CICS.
- The CICS TG supplied Java sample EciB2 available on the client machine.

Testing your TCP/IP network

At the transport layer, issue ping requests between the operating system that is hosting your CICS TG and the LPAR where your CICS server resides. The ping request response, as shown in the example below, confirms that the TCP/IP communications are working. The ping request also works if CICS TG and the CICS server are not in the same LPAR or if you are using multiple IP stacks on the same LPAR.

```
ping cicssrv2.company.com

Pinging cicssrv2.company.com [1.23.456.789] with 32 bytes of data:

Reply from 1.23.456.789: bytes=32 time<1ms TTL=61
Reply from 1.23.456.789: bytes=32 time<1ms TTL=61
Reply from 1.23.456.789: bytes=32 time<1ms TTL=61
Reply from 1.23.456.789: bytes=32 time<1ms TTL=61

Ping statistics for 1.23.456.789:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Configuring the IPIC server on CICS TG

You must define a server definition for the Gateway daemon to communicate to CICS over IPIC in remote mode.

To define a server definition for the Gateway daemon:

1. Edit the ctg.ini file and define an IPICSERVER definition for your CICS server:
   a. Set HOSTNAME to the name of the z/OS machine that hosts your CICS server.
   b. Set PORT to the port number that your CICS server uses to listen for incoming IPIC requests.

   For example:
   ```
   SECTION IPICSERVER = CICSA
       HOSTNAME=cicssrv2.company.com
       PORT=50889
   ENDSECTION
   ```
2. Save your updated ctg.ini file.
3. Edit the CTGS01A1 data set and define the configuration and Java paths:
   - Replace <config_path> with the directory that ctg.ini is stored in.

- Replace <javag_path> with your Java 7 install path. For example:

```
CICSCLI=/u/ctguser/ctg.ini
PATH=/bin:/java/java70/bin
```

4. Save your updated data set.
5. Start CICS TG to apply the new IPICSERVER definition.

Configuring the IPCONN autoinstall user program DFHISCIP on CICS TS

To enable the autoinstall of multiple secure IPCONNs, you must modify the sample IPCONN autoinstall program.

CICS provides the IPCONN autoinstall sample program called DFHISxIP in Assembler, C, COBOL, and PL/I , where 'x' denotes the language, which is A, D, C, and P respectively. The sample program does not use a template by default, so, for autoinstall requests to use a template you must update the program. In this example, the COBOL user program DFHISCIP is updated.

1. Add the MOVE statement to the autoinstall user program DFHISCIP in the A010-INSTALL-IPCONN section. This statement requests CICS to use the IPCONN template SECTEMPL each time the autoinstall user program is called.

```
* - - - - - - - - - - -
* Install processing
* - - - - - - - - - - -
 A010-INSTALL-IPCONN SECTION.
* Template for secure IPCONN
     MOVE 'SECTEMPL' TO ISAIC-TEMPLATE
```

2. Compile and link-edit your program into a data set that will be picked up by your CICS server.

Configuring the TCPIPSERVICE on CICS TS

The TCPIPSERVICE is a resource that defines the attributes of the IPIC connection, including the listening port and the IPCONN autoinstall user program, referred to as a user replaceable module (URM).

1. Use CEDA to define a TCPIPSERVICE; for example, SRV50889. These values are important:
   - The URM is set to point to your compiled IPCONN autoinstall user program.
   - The port number is set for incoming IPIC requests.
   - The protocol is set to IPIC.
   - The transaction is set to CISS.

   All other values can be left to default. The security section of the TCPIPSERVICE is not applicable for the IPIC protocol; security is applied in the IPCONN definition.

```
CEDA  DEFine TCpipservice( SRV50889 )
 TCpipservice   : SRV50889
 GROup          : HOLLTCPA
 DEscription  ==>
 Urm          ==> DFHISCIP
 POrtnumber   ==> 50889            1-65535
 STatus       ==> Open             Open | Closed
 PROtocol     ==> IPIC             IIop | Http | Eci | User | IPic
 TRansaction  ==> CISS
 Backlog      ==> 00001            0-32767
 TSqprefix    ==>
 Ipaddress    ==>
 SOcketclose  ==> No                No | 0-240000 (HHMMSS)
 Maxdatalen   ==>                   3-524288
```

2. Install the CEDA definition.
3. Check that the TCPIPSERVICE is active. On CICS TS, issue the command:

   ```
   CEMT INQ TCPIPSERVICE
   ```

   Check these values:
   - The port number shown is correct.
   - The status shows "Ope" for open.
   - The protocol shown is Ipic.
   - The URM shows the IPCONN autoinstall program that you modified.

     For example:

     ```
     CEMT INQ TCPIPSERVICE
     STATUS: RESULTS - OVERTYPE TO MODIFY
     Tcpips(SRV50889) Ope Por(50889) Ipic Nos Tra(CISS)
     Con(00000) Bac( 00001 ) Max( 000000 ) Urm( DFHISCIP )
     ```

Note: You can configure CICS resources using the CICS Explorer , see the CICS Explorer information in the CICS TS Information Center for more information.

Configuring the IPCONN template on CICS TS

You must define the IPCONN template that each incoming IPIC connection uses. This example implements both link security and user security.

1. Use CEDA to define an IPCONN. The name of the IPCONN must match the name of the template specified in the IPCONN autoinstall user program; for example, SECTEMPL. These values are important:

   TCPIPService
   Set this value to match the name of the TCPIPService defined earlier.

   Receivecount
   Set this value to specify the number of parallel IPCONN sessions.

   SENdcount
   Set this value to zero because IPIC connections are always inbound to CICS TS from CICS TG.

   Inservice
   Set this value to `Yes`.

   Linkauth
   Set this value to `Secuser`.

   SECurityname
   Set this value to an authorized RACF user ID. The user ID must be in a RACF group that is authorized to establish IPIC connections.

   Userauth
   Set this value to `Verify`.

   The APPLID field is relevant only for predefined IPCONN connections. The APPLID field is ignored for autoinstalled IPCONN connections. CICS populates this field with the name of the IPCONN by default.

   This panel is an example of an IPCONN template defined using the CEDA transaction:

   ```
   CEDA  View Ipconn( SECTEMPL )
    Ipconn         : SECTEMPL
    Group          : HOLLIPIC
    Description    :
   IPIC CONNECTION IDENTIFIERS
    APplid         : SECTEMPL
   ```

```
 Networkid      :
 Host           :
 (Lower Case)   :
 Port           : No            No | 1-65535
 Tcpipservice   : SRV50889
IPIC CONNECTION PROPERTIES
 Receivecount   : 100           1-999
 SENdcount      : 000           0-999
 Queuelimit     : No            No | 0-9999
 Maxqtime       : No            No | 0-9999
OPERATIONAL PROPERTIES
 AUtoconnect    : No            No | Yes
 Inservice      : Yes           Yes | No
SECURITY
 SSl            : No            No | Yes
 CErtificate    :                                   (Mixed Case)
 CIphers        :
 Linkauth       : Secuser       Secuser | Certuser
 SECurityname   : LINKUSER
 Userauth       : Verify        Local | Identify | Verify | Defaultuser
RECOVERY
 Xlnaction      : Keep          Keep | Force
```

2. Install the IPCONN definition and check that the output from the CEMT INQ IPCONN(SECTEMPL) command identifies it as INService RELeased.

```
CEMT I IPCONN
 STATUS:  RESULTS - OVERTYPE TO MODIFY
  Ipc(SECTEMPL) App(SECTEMPL) Net(GBIBMIYA) Ins Rel Nos
     Rece(100) Sen(000) Tcp(SRV50889)
```

Testing your scenario

To test that your scenario is configured correctly, 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.

Configuring a secure predefined IPIC connection (SC02)

A predefined IPCONN provides a more secure environment and can prevent unwanted IPCONN autoinstall requests from succeeding. To configure a secure predefined IPCONN for your IPIC connection between CICS TG and CICS TS, follow the instructions in this scenario.

This scenario uses CICS TG connecting to CICS TS V3.2 over IPIC in remote mode. It uses the default name `ctg.ini` for the configuration file.

Table 18. Values used in this scenario

Component	**Parameter**	**Where set**	**Example value**	**Matching values**
CICS TG	APPLID **1**	PRODUCT section of ctg.ini	MYAPPL	This value must be the same as **6**
CICS TG	APPLIDQUALIFIER **2**	PRODUCT section of ctg.ini	MYQUAL	This value must be the same as **7**
CICS TG	Server name	IPICSERVER section of ctg.ini	CICSA	
CICS TG	Hostname	IPICSERVER section of ctg.ini	cicssrv2.company.com	
CICS TG	Port **3**	IPICSERVER section of ctg.ini	50889	This value must be the same as **5**
CICS TS	TCPIPService **4**	TCPIPService definition	SRV50889	This value must be the same as **8**

Table 18. Values used in this scenario (continued)

Component	Parameter	Where set	Example value	Matching values
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	

Prerequisites

You must satisfy these system requirements.

Here are the system requirements for CICS TS for z/OS:

- The server must be CICS V3.2 or later because IPIC is not available in earlier releases of CICS.
- TCP/IP services must be active in the CICS server.
  - To activate these services, set the TCP system initialization parameter to YES.
  - To check the status of these services, issue a CEMT INQ TCPIP command and check that the status is open.
- The CICS server must have access to a TCP/IP stack running on the same LPAR.
- The TCP/IP network must extend between LPARs if CICS TG for z/OS and the CICS server exist on different LPARs.
- You must set the SEC system initialization parameter to YES to enable security.
- You must have valid RACF user IDs and passwords.

Here are the system requirements for CICS TG:

- CICS TG must be installed.

To test that the scenario works successfully you can use either the supplied samples, or your own applications. If you use the supplied samples, this scenario requires the following:

- The sample CICS TG server program EC01 must be compiled, defined, and installed on CICS.

- The CICS TG supplied Java sample EciB2 available on the client machine.

Testing your TCP/IP network

At the transport layer, issue ping requests between the operating system that is hosting your CICS TG and the LPAR where your CICS server resides. The ping request response, as shown in the example, confirms that the TCP/IP communications are working. The ping request also works if CICS TG and the CICS server are not in the same LPAR or 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. Edit the CTGS02A1 data set and define the configuration and Java paths:
   a. Replace <config_path> with the directory that ctg.ini is stored in.
   b. Replace <java_path> with your Java 7 install path. For example:

      ```
      CICSCLI=/u/ctguser/ctg.ini
      PATH=/bin:/java/java70/bin
      ```

5. Save the updated CTGS02A1 data set.
6. Start CICS TG to apply the new definitions.

Configuring the TCPIPService on CICS TS

The TCPIPService is a resource that defines the attributes of the IPIC connection, including the listening port.

1. Use CEDA to define a TCPIPService; for example, SRV50889. These values are important:
   - The URM is set to NO to prevent the default IPCONN autoinstall program from running.
   - The port number is set for incoming IPIC requests.
   - The protocol is set to IPIC.
   - The transaction is set to CISS.

   All other values can be left to default. The security section of the TCPIPService is not applicable for the IPIC protocol; security is applied in the IPCONN definition.

   ```
   CEDA  DEFine TCpipservice( SRV50889  )
          TCpipservice  : SRV50889
          GROup          : HOLLIPIC
          DEscription  ==>
          Urm          ==> NO
          POrtnumber   ==> 50889     1-65535
          STatus       ==> Open      Open | Closed
          PROtocol     ==> IPIC      IIop | Http | Eci | User | IPic
          TRansaction  ==> CISS
          Backlog      ==> 00001     0-32767
          TSqprefix    ==>
          Ipaddress    ==>
          SOcketclose  ==> No                 No | 0-240000 (HHMMSS)
          Maxdatalen   ==>                    3-524288
   ```
2. Install the CEDA definition.
3. Check that the TCPIPService is active. On CICS TS, issue the command:

   ```
   CEMT INQ TCPIPSERVICE
   ```

   Check the following values:
   - The port number shown is correct.
   - The status shows "Ope" for open.
   - The protocol shown is Ipic.
   - The URM shows NO to state that IPCONN autoinstall is not permitted on this TCPIPSERVICE.

     For example:

     ```
     CEMT INQ TCPIPSERVICE
     STATUS: RESULTS - OVERTYPE TO MODIFY
     Tcpips(SRV50889) Ope Por(50889) Ipic Nos Tra(CISS)
     Con(00000) Bac( 00001 ) 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

      CICS servers defined:
      System : CICSA

      Call Programs
      About to call : EC01
        Commarea    :
        Extend_Mode : 0
        Luw_Token   : 0
        Commarea    : 24/06/09 11:17:19
      Return code   : ECI_NO_ERROR(0)
      Abend code    : null
      Successfully closed JavaGateway
```

   In the CICS job log you will see this message:

```
DFHIS2001 ... Client session from applid MYAPPL accepted for
IPCONN IPC50889.
```

   Issuing CEMT INQ TCPIPSERVICE shows that the connection count has increased to 1.

```
CEMT INQ TCPIPSERVICE
         STATUS: RESULTS - OVERTYPE TO MODIFY
          Tcpips(SRV50889) Ope Por(50889) Ipic Nos Tra(CISS)
           Con(00001) Bac( 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.
```

Configuring a highly available Gateway group with two-phase commit and IPIC (SC03)

This scenario shows how to configure a highly available gateway group (HA group) of Gateway daemons with connections to CICS over IPIC. The scenario supports two-phase commit (XA) transactions and uses policy-based dynamic server selection for selecting a CICS server.

A highly available gateway group consists of a group of Gateway daemons that share the same TCP/IP port and are capable of providing XA transactional support.

The figure shows workload from a connection factory in WebSphere Application Server being served by similar Gateway daemons (CTGA1, CTGA2) which represent the HA group “GroupA”. Any supported JEE application server can be used in this scenario.

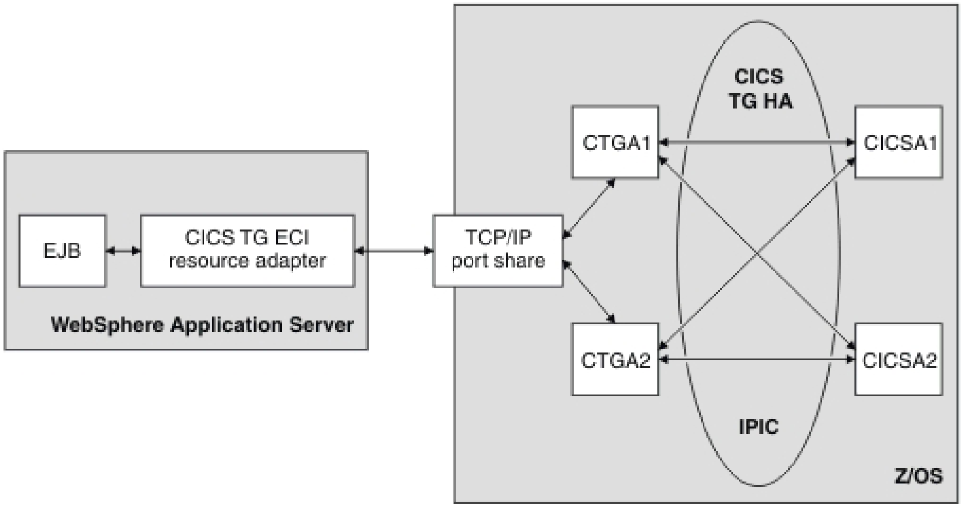


Figure 22. XA transactions over IPIC in a high availability scenario

Each Gateway daemon is connected to CICS servers CICSA1 and CICSA2 using the IPIC protocol. A transaction might be handled by Gateway daemon CTGA1 or CTGA2, and 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 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 an HA group. This example uses CICS Transaction Gateway connecting to CICS Transaction Server V4.1 over IPIC and WebSphere Application Server V8.0.

Table 19. Values used in this scenario

Comp- onent	Property	Where set	Details
TCP/IP	Port sharing	PROFILE.TCPIP	4148 TCP CTGA* SHAREPORT
CICS TG	APPLIDQUALIFIER	GroupA_GW.ini	GROUPA
CICS TG	TCP protocol handler	GroupA_GW.ini	Port number 4148

CICS TG	Gateway group A common configuration	GroupA_GW.ini	maxconnect=500
			maxworker=250
			xasupport=on

CICS TG	IPIC connection to CICSA1	GroupA_GW.ini	name=CICSA1
			hostname=server.ibm.com
			port=4149

Table 19. Values used in this scenario (continued)

Comp- onent	Property	Where set	Details

CICS TG	IPIC connection to CICSA2	GroupA_GW.ini	name=CICSA2
			hostname=server.ibm.com
			port=4150

CICS TG A1	Gateway instance A1 environment variables	CTGS03A1	CICSCLI=*config_path*/GroupA_GW.ini
			CTGSTART_OPTS=-applid=CTGA1
			-statsport=4151

CICS TG A2	Gateway instance A2 environment variables	CTGS03A2	CICSCLI=*config_path*/GroupA_GW.ini
			CTGSTART_OPTS=-applid=CTGA2
			-statsport=4152

CICS TS CICSA1	TCPIPService for IPIC connection	Using CEDA on CICSA1	name=SRV4149
			protocol=IPIC
			port=4149

CICS TS CICSA2	TCPIPService for IPIC connection	Using CEDA on CICSA2	name=SRV4150
			protocol=IPIC
			port=4150

WebSphere Application Server	CICS TG ECI resource adapter connection factory	J2C connection factories	name=ECI-XA-GROUPA
WebSphere Application Server	CICS TG ECI resource adapter connection factory	J2C connection factories	JNDI Name=eis/CICSGroupA
WebSphere Application Server	CICS TG ECI resource adapter connection factory	ECI-XA-GROUPA connection pool properties	Maximum connections=500

WebSphere Application Server	CICS TG ECI resource adapter connection factory	ECI-XA-GROUPA custom properties	ConnectionURL=tcp://server.ibm.com
			PortNumber=4148
			ApplidQualifier=GROUPA
			Applid=XAWASA
			xasupport=on

Prerequisites

The prerequisites for CICS Transaction Gateway, CICS Transaction Server and WebSphere Application Server.

Here are the system requirements for CICS Transaction Server for 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.
- The TCP/IP network must extend between LPARs if CICS Transaction Gateway for z/OS and the CICS server exist on different LPARs.

Here are the system requirements for CICS Transaction Gateway:

- CICS Transaction Gateway must be installed.

- All prerequisites for XA support must be satisfied. For more information see “Configuring for XA transaction support” on page 126

Here are the system requirements for WebSphere Application Server:

- The Installation Verification Test (IVT) that you run to test this scenario uses the ECI resource adapter, cicseci.rar with XA support enabled. The resource adapter must be downloaded onto the machine that runs the Web browser you use for accessing the 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 165.

Configuring CICS TG for high availability

Configuring CICS Transaction Gateway for high availability requires the creation of various JCL files and configuration files.

The common configuration file is installed in the HFS under the directory <install_path>/samples/scenarios/sc03. The sample environment files are available in the partitioned data set (PDS) *install_HLQ*.SCTGSAMP.

The configuration files used to define an HA group GROUPA are:

- A common CICS Transaction Gateway configuration file
- An individual file for each Gateway daemon containing environment variables (used in the CTGBATCH job step STDENV DD definition)

CICS Transaction Gateway configuration file

Create a CICS Transaction Gateway configuration file GroupA_GW.ini using the values suggested in the table Table 19 on page 191:

1. Define the HA group GROUPA; define the APPLIDQUALIFIER as GROUPA.

   For example:

```
SECTION PRODUCT
   APPLIDQUALIFIER=GROUPA # Common group APPLID qualifier
                          # APPLID is overridden by CTGSTART_OPTS
ENDSECTION
```

2. Define the common Gateway daemon characteristics. Edit the GroupA_GW.ini file:
   a. Define the thread pool sizes for connection managers and worker.
   b. Enable dynamic server selection policy.
   c. Enable XA support.
   d. Define the TCP/IP protocol handler using port number 4148.
   e. Define a statistics protocol handler. The port number will be overridden by the `ctgstart` switch `-statsport`.

   For example:

```
SECTION GATEWAY
    maxconnect=500
    maxworker=250
    noinput=on
    xasupport=on

# Enable dynamic server selection policy
    DSSPOLICY=POLICY1
```

```
# Define tcp protocol handler, port 4148
    protocol@tcp.handler=com.ibm.ctg.server.TCPHandler
    protocol@tcp.parameters=connecttimeout=2000;
                            idletimeout=0;\
                            pingfrequency=10000;
                            port=4148;
                            bind=;
                            solinger=0;

# Define stats protocol handler
# Note: This port number is overridden by CTGSTART_OPTS
    protocol@statsapi.handler=com.ibm.ctg.server.RestrictedTCPHandler
    protocol@statsapi.parameters=port=;
                                 bind=;
                                 connecttimeout=2000;
                                 maxconn=5;
ENDSECTION
```

3. 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 GroupA_GW.ini file and add an IPICSERVER section for each of your CICS servers:
   - Set HOSTNAME to the name of the 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
CONNECTTIMEOUT=5
SENDSESSIONS=250
ENDSECTION

SECTION IPICSERVER = CICSA2
DESCRIPTION=IPIC connection to CICSA2
HOSTNAME=server.ibm.com
PORT=4150
CONNECTTIMEOUT=5
SENDSESSIONS=250
ENDSECTION
```

   CONNECTTIMEOUT applies when the target machine is not reachable. For example, if the target LPAR is down or TCP/IP on the target LPAR has been shut down.
4. You must define a dynamic server selection policy and associated dynamic server selection group which will be used to determine which CICS server requests are sent to.

   Edit the GroupA_GW.ini file and add a DSSPOLICY and DSSGROUP section.

   Set Servers to the name of the CICS servers to contact. Set Algorithm to the algorithm to be used for choosing a CICS server.

   For example:

```
# Define dynamic server selection policy
SECTION DSSPOLICY = POLICY1
SUBSECTION MAPPINGS
<NONE>=GROUP1
ENDSUBSECTION
ENDSECTION

# Define dynamic server selection group
```

```
SECTION DSSGROUP = GROUP1
Servers=CICSA1,CICSA2
Algorithm=ROUNDROBIN
ENDSECTION
```

CICS Transaction Gateway environment variables

Define the characteristics specific to Gateway daemon CTGA1. Create the file CTGS03A1 with the following environment variables:

```
CICSCLI=<config_path>/GroupA_GW.ini
CTGSTART_OPTS=-applid=CTGA1 -statsport=4151
PATH=/bin:/usr/sbin:<java_path>/bin
CTG_EXCI_INIT=NO
_BPX_SHAREAS=YES
```

Define the individual Gateway daemon characteristics for Gateway daemon CTGA2. Create the file CTGS03A2 with the following environment variables:

```
CICSCLI=<config_path>/GroupA_GW.ini
CTGSTART_OPTS=-applid=CTGA2 -statsport=4152
PATH=/bin:/usr/sbin:<java_path>/bin
CTG_EXCI_INIT=NO
_BPX_SHAREAS=YES
```

Configuring TCP/IP for port sharing

Configure TCP/IP for port sharing by adding a PORT entry to the TCPIP profile, to associate the shared port number with the corresponding Gateway daemon job names CTGA1, CTGA2.

Add the following PORT entry to the TCPIP profile:

```
4148 TCP CTGA* SHAREPORT
```

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 servers require separate TCPIPService definitions, using the port numbers defined in the table of values in “Configuring a highly available Gateway group with two-phase commit and IPIC (SC03)” on page 190.

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      ==> 00001           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 and 4150 for CICSA2.
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( 00001 ) Max( 000000 ) Urm( DFHISCIP )
     ```

Configuring WebSphere Application Server

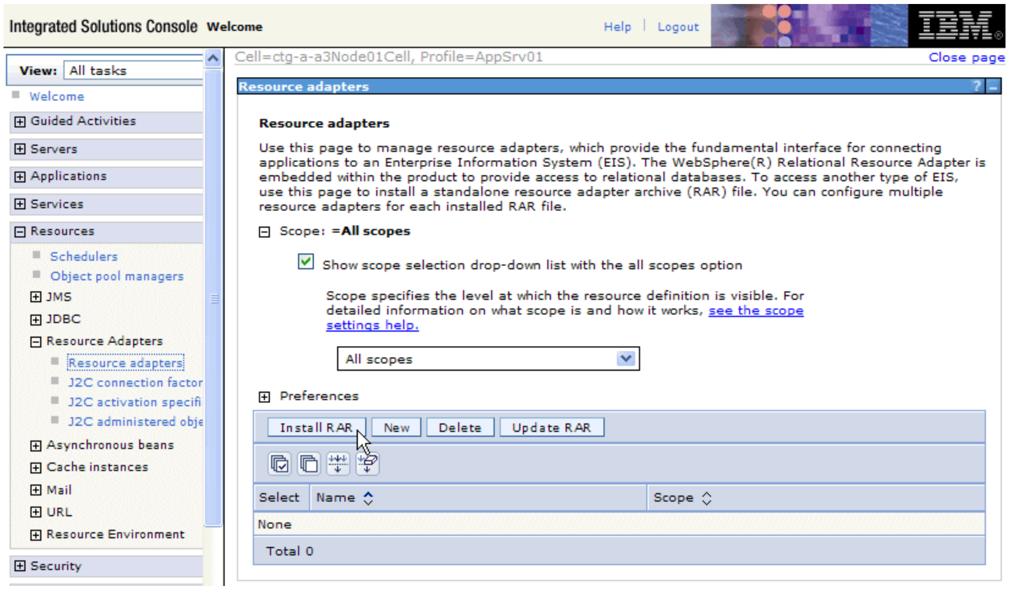
Use the WebSphere Application Server Integrated Solutions Console (the “admin console”) to install and configure the CICS ECI resource adapter (cicseci.rar) and enable XA support.

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 23 on page 197.



Cell=ctg-a-a3Node01Cell, Profile=AppSrv01

Resource adapters

Use this page to manage resource adapters, which provide the fundamental interface for connecting applications to an Enterprise Information System (EIS). The WebSphere(R) Relational Resource Adapter is embedded within the product to provide access to relational databases. To access another type of EIS, use this page to install a standalone resource adapter archive (RAR) file. You can configure multiple resource adapters for each installed RAR file.

Scope: **=All scopes**

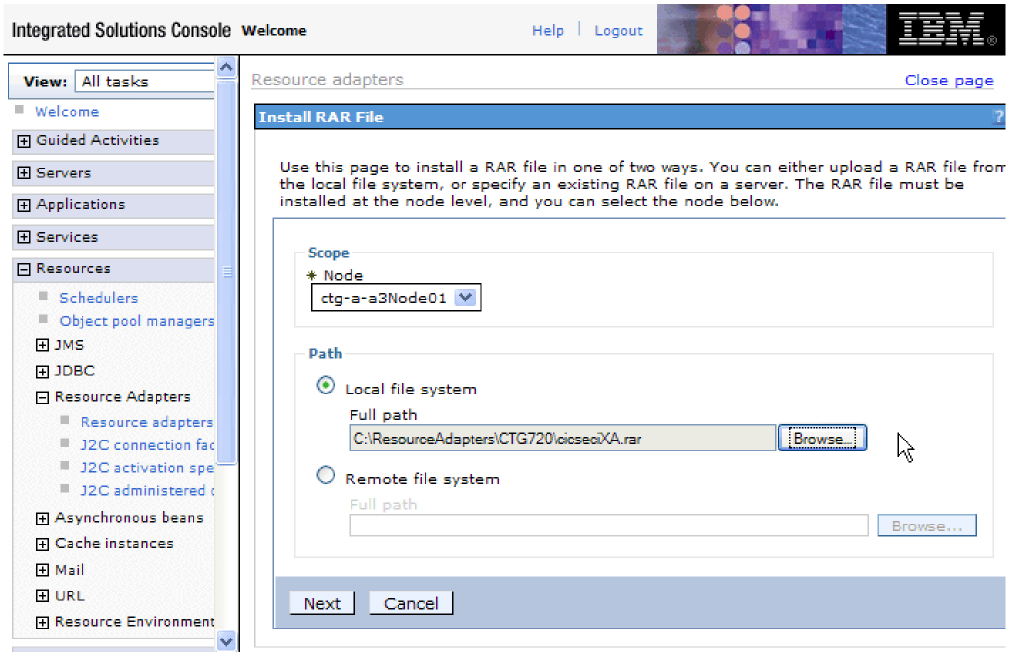
☑ Show scope selection drop-down list with the all scopes option

Scope specifies the level at which the resource definition is visible. For detailed information on what scope is and how it works, see the scope settings help.

All scopes

Figure 23. Installing a new resource adapter

From the **Install RAR file** dialog shown in Figure 24, click **Browse** for the Remote 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*.



Use this page to install a RAR file in one of two ways. You can either upload a RAR file from the local file system, or specify an existing RAR file on a server. The RAR file must be installed at the node level, and you can select the node below.

Scope

* Node

ctg-a-a3Node01

Path

● Local file system

Full path

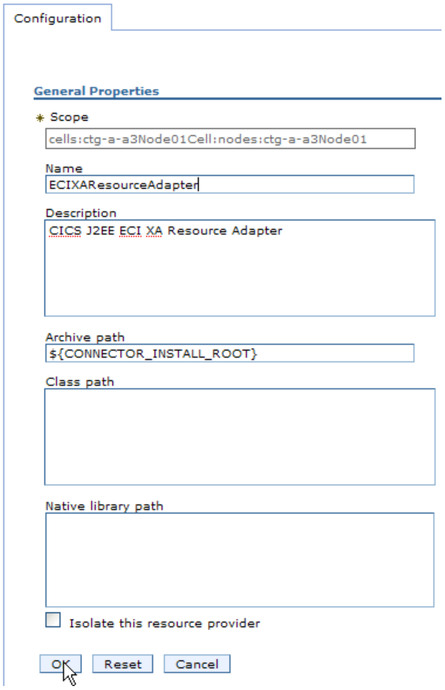
C:\ResourceAdapters\CTG720\cicseciXA.rar

○ Remote file system

Full path

Figure 24. Selecting the ECI RAR file for installation

Click **Next**. The resource adapter **General Properties** dialog shown in Figure 25 on page 198 contains a predefined name and description for the CICS ECI resource adapter.



* Scope
cells:ctg-a-a3Node01Cell:nodes:ctg-a-a3Node01

Name
ECIXAResourceAdapter

Description
CICS J2EE ECI XA Resource Adapter

Archive path
${CONNECTOR_INSTALL_ROOT}

Class path

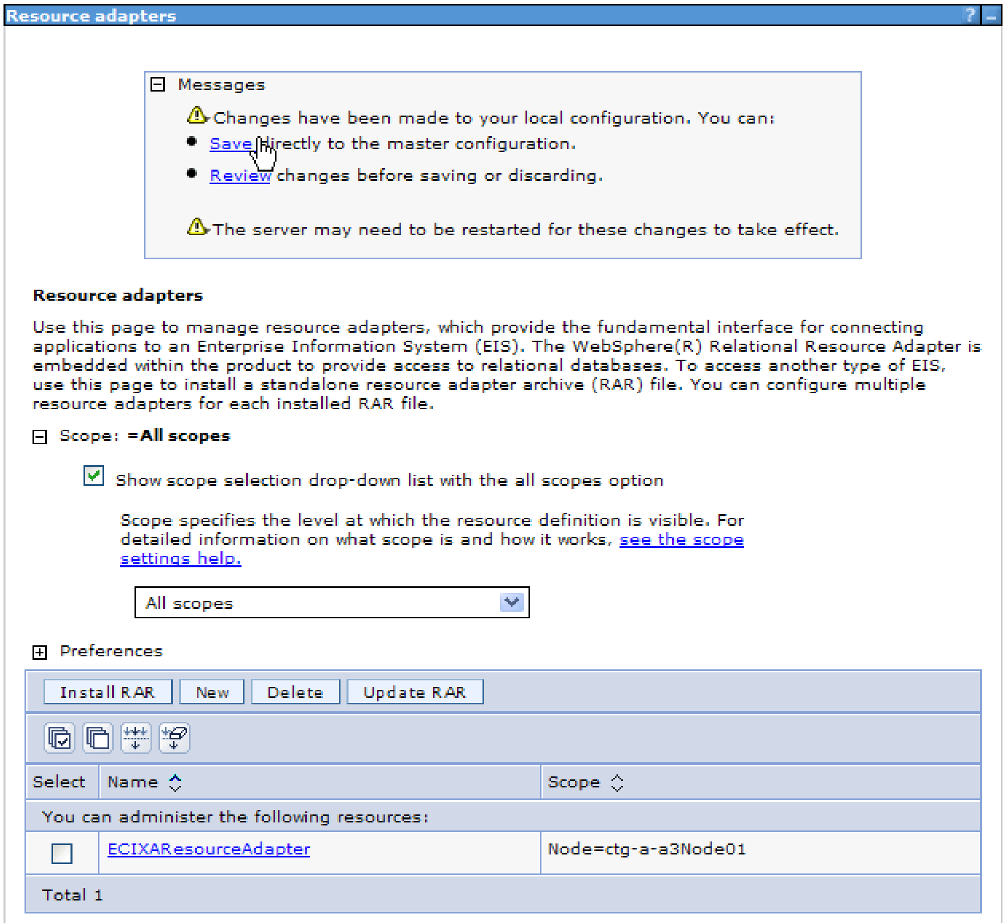
Native library path

☐ Isolate this resource provider

Figure 25. 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 26 on page 199, 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.



Messages

Changes have been made to your local configuration. You can:

- Save directly to the master configuration.
- Review changes before saving or discarding.

The server may need to be restarted for these changes to take effect.

Resource adapters

Use this page to manage resource adapters, which provide the fundamental interface for connecting applications to an Enterprise Information System (EIS). The WebSphere(R) Relational Resource Adapter is embedded within the product to provide access to relational databases. To access another type of EIS, use this page to install a standalone resource adapter archive (RAR) file. You can configure multiple resource adapters for each installed RAR file.

Scope: =**All scopes**

☑ Show scope selection drop-down list with the all scopes option

Scope specifies the level at which the resource definition is visible. For detailed information on what scope is and how it works, see the scope settings help.

All scopes

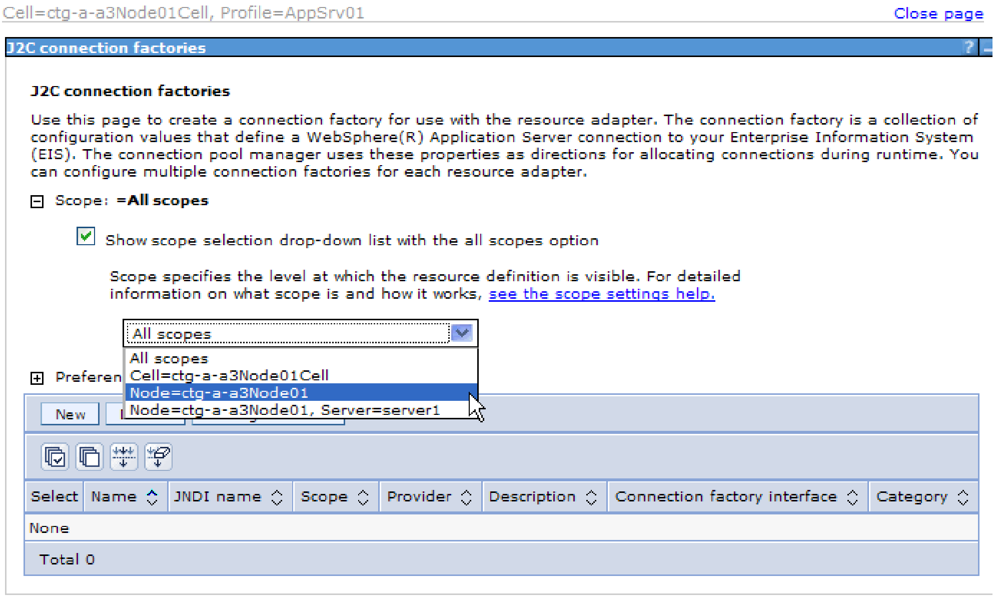
Select	Name	Scope
You can administer the following resources:		
☐	ECIXAResourceAdapter	Node=ctg-a-a3Node01
Total 1		

Figure 26. 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 27 on page 200. 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.



Cell=ctg-a-a3Node01Cell, Profile=AppSrv01

J2C connection factories

Use this page to create a connection factory for use with the resource adapter. The connection factory is a collection of configuration values that define a WebSphere(R) Application Server connection to your Enterprise Information System (EIS). The connection pool manager uses these properties as directions for allocating connections during runtime. You can configure multiple connection factories for each resource adapter.

⊟ Scope: =**All scopes**

☑ Show scope selection drop-down list with the all scopes option

Scope specifies the level at which the resource definition is visible. For detailed information on what scope is and how it works, see the scope settings help.

All scopes

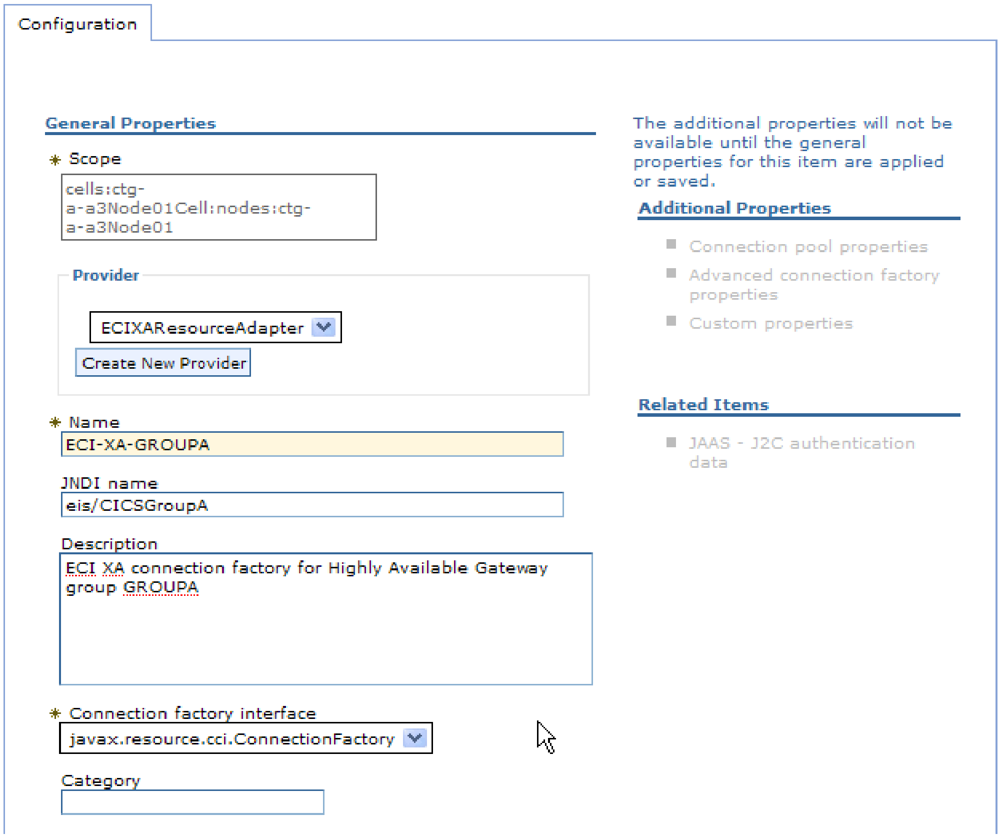
Select	Name	JNDI name	Scope	Provider	Description	Connection factory interface	Category
None							
Total 0							

Figure 27. 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 “Configuring a highly available Gateway group with two-phase commit and IPIC (SC03)” on page 190, and are shown in the 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.


Configuration
General Properties
The additional properties will not be available until the general properties for this item are applied or saved.
Additional Properties
Connection pool properties
Advanced connection factory properties
Custom properties
Related Items
JAAS - J2C authentication data
Create New Provider

Scope: cells:ctg-a-a3Node01Cell:nodes:ctg-a-a3Node01

Provider: ECIXAResourceAdapter

Name: ECI-XA-GROUPA

JNDI name: eis/CICSGroupA

Description: ECI XA connection factory for Highly Available Gateway group GROUPA

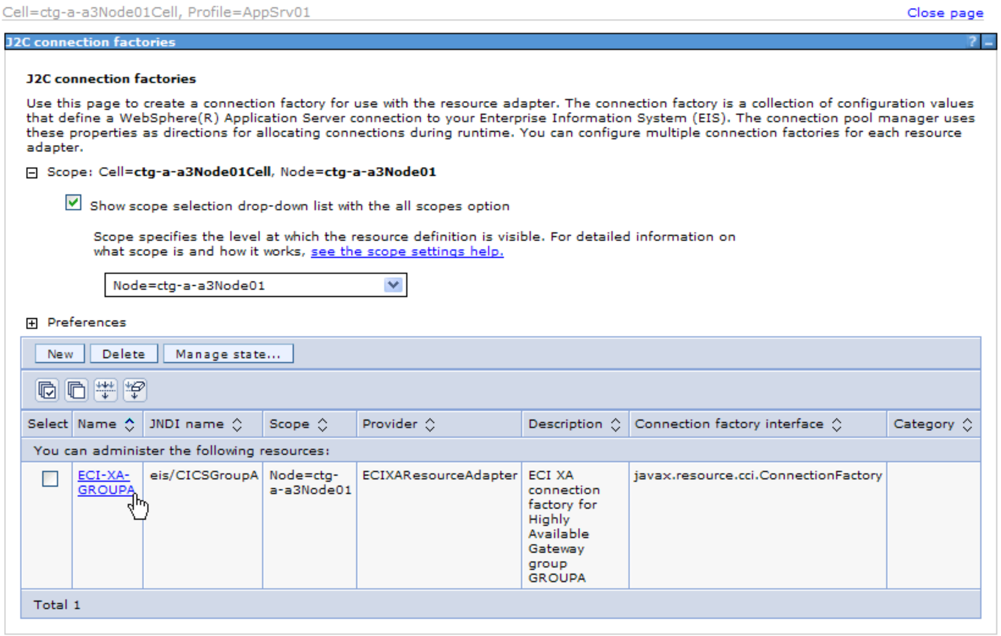
Connection factory interface: javax.resource.cci.ConnectionFactory

Category:

Figure 28. 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 29 on page 202.



Cell=ctg-a-a3Node01Cell, Profile=AppSrv01

J2C connection factories

J2C connection factories

Use this page to create a connection factory for use with the resource adapter. The connection factory is a collection of configuration values that define a WebSphere(R) Application Server connection to your Enterprise Information System (EIS). The connection pool manager uses these properties as directions for allocating connections during runtime. You can configure multiple connection factories for each resource adapter.

⊟ Scope: Cell=**ctg-a-a3Node01Cell**, Node=**ctg-a-a3Node01**

☑ Show scope selection drop-down list with the all scopes option

Scope specifies the level at which the resource definition is visible. For detailed information on what scope is and how it works, see the scope settings help.

Node=ctg-a-a3Node01

⊞ Preferences

Select	Name	JNDI name	Scope	Provider	Description	Connection factory interface	Category
You can administer the following resources:							
☐	ECI-XA-GROUPA	eis/CICSGroupA	Node=ctg-a-a3Node01	ECIXAResourceAdapter	ECI XA connection factory for Highly Available Gateway group GROUPA	javax.resource.cci.ConnectionFactory	
Total 1							

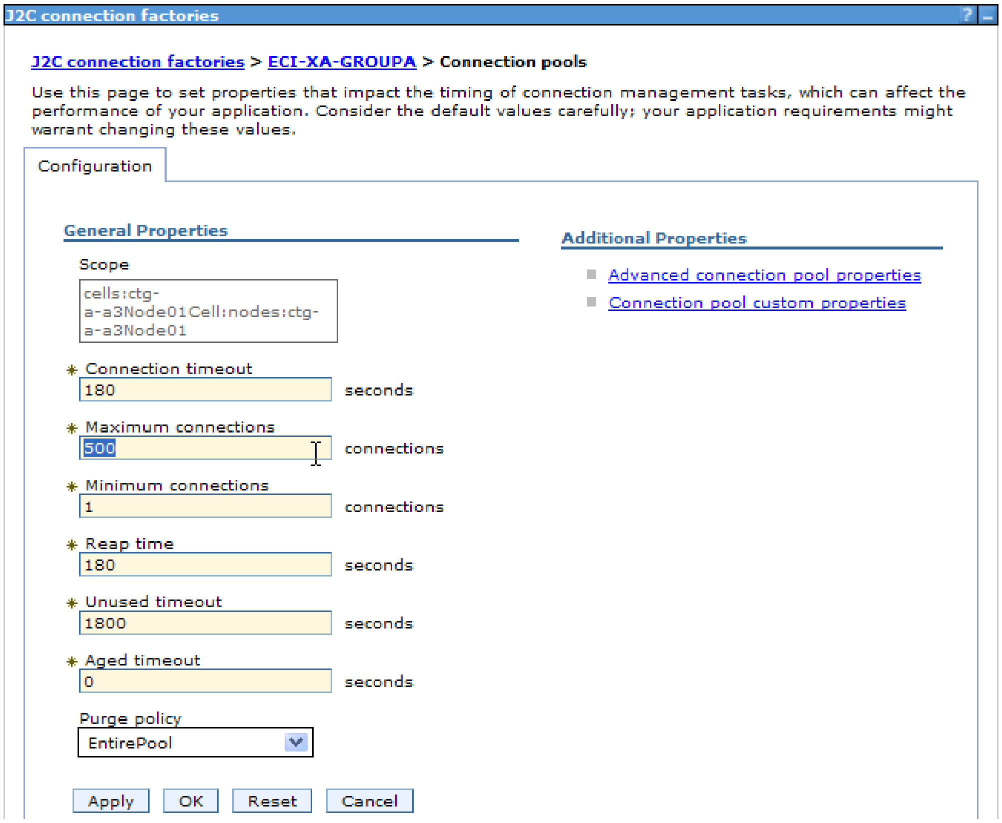
Figure 29. The new J2C connection factory

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-XA-GROUPA from the table of connection factories shown in Figure 30 on page 203.

The admin console returns you to the **J2C connection factories - General Properties** dialog for the connection factory ECI-XA-GROUP. From the **Additional properties** section, select **Connection pool properties**. The admin console displays the **Connection pool properties** configuration dialog shown in Figure 30 on page 203.



J2C connection factories > ECI-XA-GROUPA > Connection pools

Use this page to set properties that impact the timing of connection management tasks, which can affect the performance of your application. Consider the default values carefully; your application requirements might warrant changing these values.

General Properties

Scope: cells:ctg-a-a3Node01Cell:nodes:ctg-a-a3Node01

* Connection timeout: 180 seconds

* Maximum connections: 500 connections

* Minimum connections: 1 connections

* Reap time: 180 seconds

* Unused timeout: 1800 seconds

* Aged timeout: 0 seconds

Purge policy: EntirePool

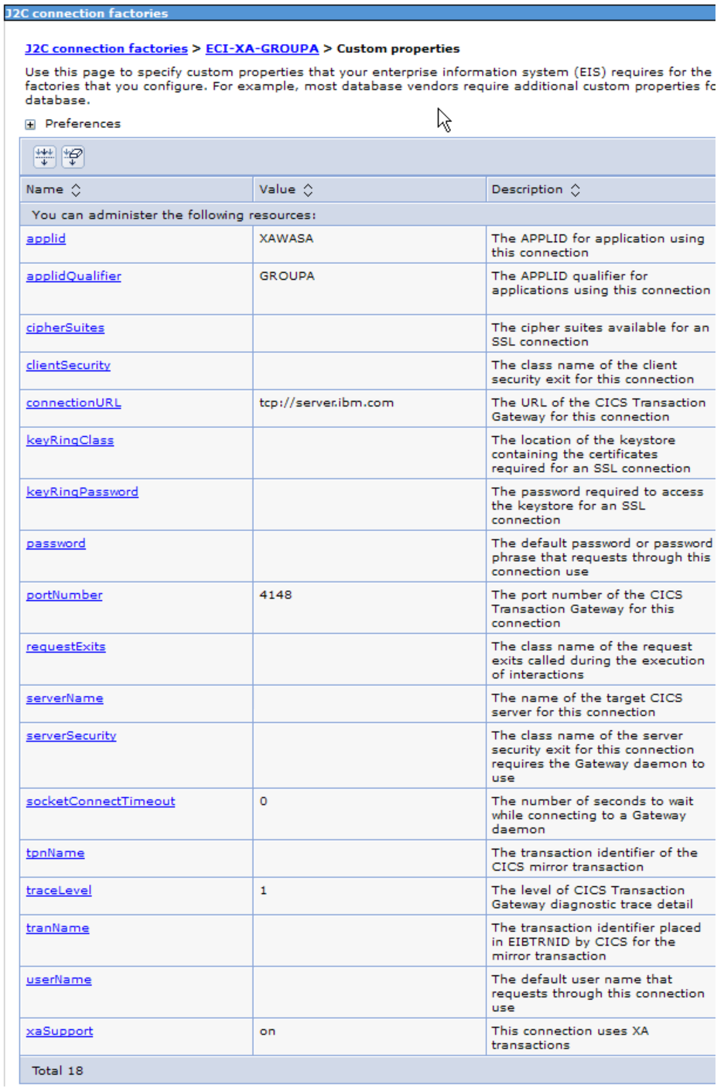
Figure 30. Configuring the connection pools

Change the value of *Maximum connections* to 500. If TCP/IP load balancing results in an even distribution of connections, each Gateway daemon will have 250 connections. If one Gateway daemon should become unavailable, the other Gateway must be able to handle 500 connections. This is the reason for configuring the Gateway daemons with MAXCONNECT=500.

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 31 on page 204.



J2C connection factories

J2C connection factories > ECI-XA-GROUPA > **Custom properties**

Use this page to specify custom properties that your enterprise information system (EIS) requires for the factories that you configure. For example, most database vendors require additional custom properties fc database.

Name	Value	Description
You can administer the following resources:		
applid	XAWASA	The APPLID for application using this connection
applidQualifier	GROUPA	The APPLID qualifier for applications using this connection
cipherSuites		The cipher suites available for an SSL connection
clientSecurity		The class name of the client security exit for this connection
connectionURL	tcp://server.ibm.com	The URL of the CICS Transaction Gateway for this connection
keyRingClass		The location of the keystore containing the certificates required for an SSL connection
keyRingPassword		The password required to access the keystore for an SSL connection
password		The default password or password phrase that requests through this connection use
portNumber	4148	The port number of the CICS Transaction Gateway for this connection
requestExits		The class name of the request exits called during the execution of interactions
serverName		The name of the target CICS server for this connection
serverSecurity		The class name of the server security exit for this connection requires the Gateway daemon to use
socketConnectTimeout	0	The number of seconds to wait while connecting to a Gateway daemon
tpnName		The transaction identifier of the CICS mirror transaction
traceLevel	1	The level of CICS Transaction Gateway diagnostic trace detail
tranName		The transaction identifier placed in EIBTRNID by CICS for the mirror transaction
userName		The default user name that requests through this connection use
xaSupport	on	This connection uses XA transactions
Total 18		

Figure 31. Configuring the connection factory custom properties

Set each of the custom properties individually, by clicking on the name:

Applid

ApplidQualifier

ConnectionURL

PortNumber

In this scenario the ServerName is deliberately left blank because the DSSPOLICY defined in the ctg.ini configuration file is expecting a blank server name.

xaSupport

The required values for these properties are provided in the table in “Configuring a highly available Gateway group with two-phase commit and IPIC (SC03)” on page 190; 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 highly available Gateway group scenario, you configure, then run the installation verification test (IVT) supplied with CICS Transaction Gateway.

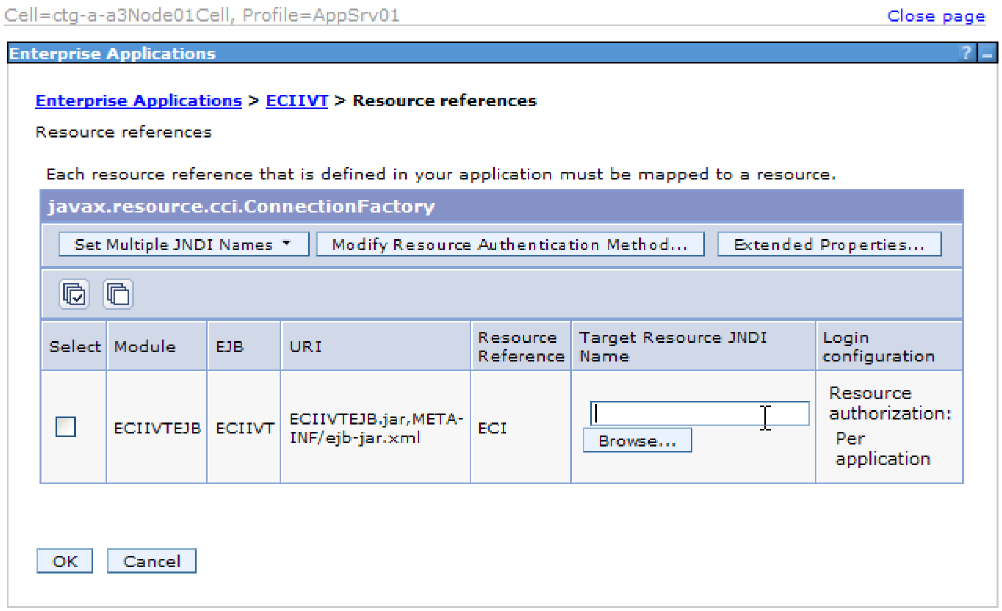
Step 1. Configure the IVT

Install the IVT application ECIIVT.ear. For more information see “JCA resource adapter installation verification test (IVT)” on page 165.

Configure the IVT application to use the highly available Gateway group GroupA, 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**.



Enterprise Applications > ECIIVT > **Resource references**

Resource references

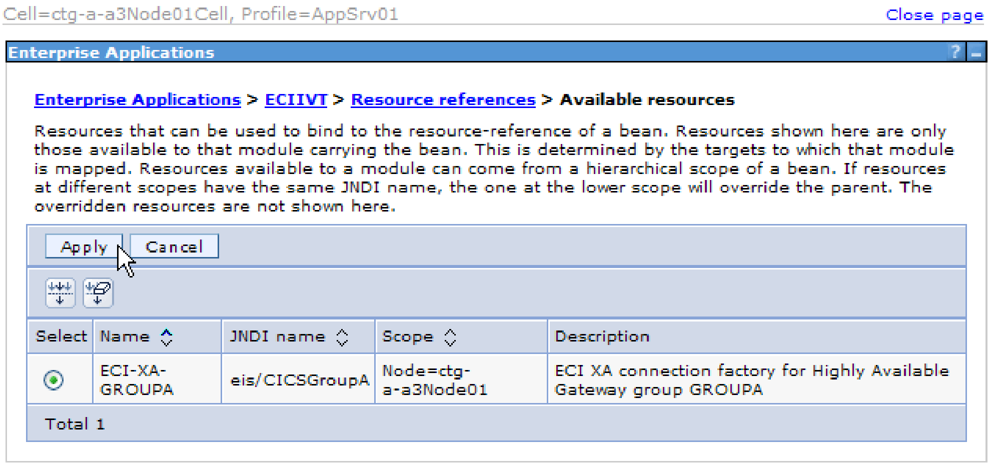
Each resource reference that is defined in your application must be mapped to a resource.

javax.resource.cci.ConnectionFactory

Select	Module	EJB	URI	Resource Reference	Target Resource JNDI Name	Login configuration
☐	ECIIVTEJB	ECIIVT	ECIIVTEJB.jar,META-INF/ejb-jar.xml	ECI		Resource authorization: Per application

Figure 32. ECI IVT resource references

The **Resource references** dialog shown in Figure 32 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 33 on page 206.



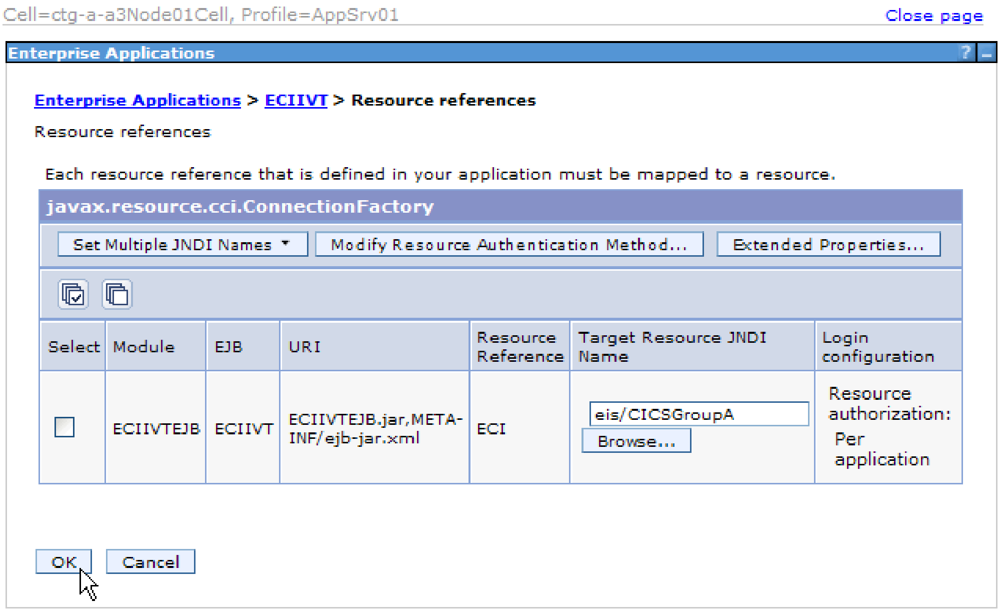
Enterprise Applications > ECIIVT > Resource references > **Available resources**

Resources that can be used to bind to the resource-reference of a bean. Resources shown here are only those available to that module carrying the bean. This is determined by the targets to which that module is mapped. Resources available to a module can come from a hierarchical scope of a bean. If resources at different scopes have the same JNDI name, the one at the lower scope will override the parent. The overridden resources are not shown here.

Select	Name	JNDI name	Scope	Description
●	ECI-XA-GROUPA	eis/CICSGroupA	Node=ctg-a-a3Node01	ECI XA connection factory for Highly Available Gateway group GROUPA
Total 1				

Figure 33. Available resources

Select ECI-XA-GROUPA resource and click **Apply**. The admin console returns you to the **Resource references** dialog. The **Target Resource JNDI Name** now has the value *eis/CICSGroupA* shown in Figure 34.



Enterprise Applications > ECIIVT > **Resource references**

Resource references

Each resource reference that is defined in your application must be mapped to a resource.

javax.resource.cci.ConnectionFactory

Select	Module	EJB	URI	Resource Reference	Target Resource JNDI Name	Login configuration
☐	ECIIVTEJB	ECIIVT	ECIIVTEJB.jar,META-INF/ejb-jar.xml	ECI	eis/CICSGroupA	Resource authorization: Per application

Figure 34. Configured ECI IVT resource references

You have now configured the ECIIVT application to use the highly available Gateway group GroupA. Save the changes to the master configuration.

Step 2. Run the IVT

Start the IVT application in WebSphere Application Server and start the gateway daemons CTGA1 and CTGA2.

Run the ECI IVT several times to send ECI requests to CICS, using different combinations of available Gateway daemons and CICS servers each time. If at least one of the CTGA1, CTGA2 Gateway daemons and 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`

If an error occurs

If an error message indicates that the connection to CICS Transaction Gateway failed, this is usually because a Gateway daemon has been shut down. The managed connections in the connection factory pool are reused but if a managed connection was connected to a Gateway daemon that is no longer running, it attempts to reuse that Gateway daemon, resulting in the error.

The solution is to ensure that the purge policy for the connection factory pool is set so that the entire connection pool is purged not just the single connection. Reset the purge policy then run the IVT again. When you have done this, if a managed connection cannot connect, all managed connections to that Gateway daemon are deallocated and the IVT should run successfully.

This error is related to the fact that high availability uses TCP/IP port sharing. The error can occur if you shut down a Gateway daemon, but should not occur if you shut down a CICS server.

Configuring identity propagation for a remote mode topology (SC04)

In this scenario, CICS Transaction Gateway and CICS Transaction Server are both on z/OS. User security information (the distributed identity) is held in IBM Tivoli Directory Server and, when it is passed to CICS Transaction Server, the identity is mapped to a user ID in RACF.

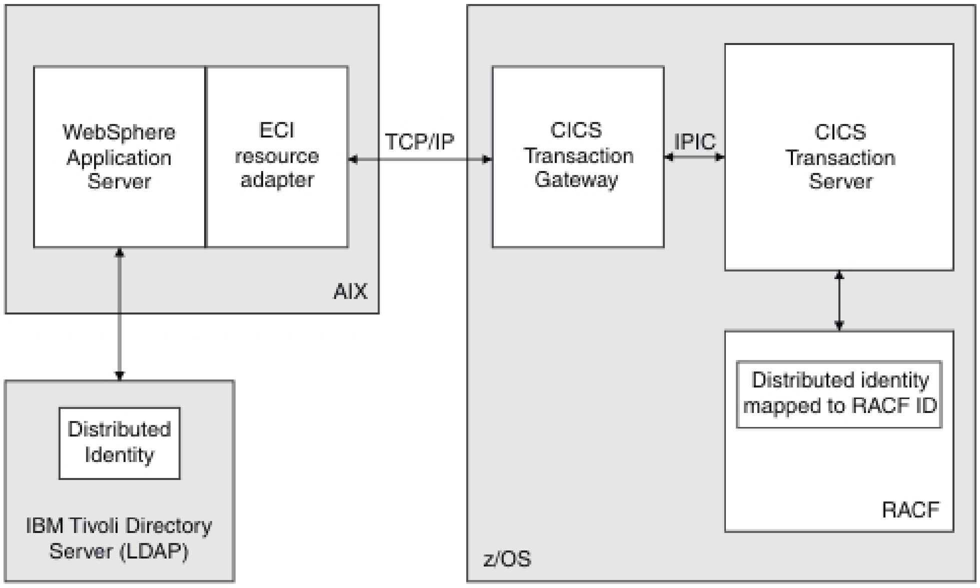


Figure 35. Topology used in this identity propagation scenario

This scenario uses WebSphere Application Server and the CICS Transaction Gateway ECI resource adapter on AIX. The CICS Transaction Gateway configuration file has the default name ctg.ini.

Values used in this scenario

Component	**Parameter**	**Where set**	**Example value**
WebSphere Application Server	Application security	WebSphere Admin Console	Enable application security (check box)
WebSphere Application Server	Authentication method	WebSphere Admin Console	CTG_idprop (the name of the identity propagation login module)
CICS TG	APPLID	PRODUCT section of ctg.ini	`MYAPPL`
CICS TG	APPLIDQUALIFIER	PRODUCT section of ctg.ini	`MYQUAL`
CICS TG	Server name	IPICSERVER section of ctg.ini	CICSA
CICS TG	HOSTNAME	IPICSERVER section of ctg.ini	`cicssrv2.company.com`
CICS TG	PORT	IPICSERVER section of ctg.ini	`50889`
CICS TS	TCPIPService	TCPIPService definition	`IPICSRV` (must match the TCPIPService specified in the IPCONN definition in CICS)
CICS TS	Portnumber	TCPIPService definition	`50889` (must match the IPICSERVER PORT specified in the ctg.ini file)
CICS TS	APplid	IPCONN definition on the CICS server	`MYAPPL` (must match the `APPLID` specified in the ctg.ini file)
CICS TS	Networkid	IPCONN definition on the CICS server	`MYQUAL` (must match the `APPLIDQUALIFIER` specified in the ctg.ini file).
CICS TS	TCPIPService	IPCONN definition on the CICS server	`IPICSRV` (must match the name of the TCPIPService in CICS)
CICS TS	Userauth	IPCONN definition on the CICS server	Must be set to `Identify`
CICS TS	IPConn	IPCONN definition on the CICS server	`IPICIP`
RACF	USERID	RACF resource access list	`TESTID`
RACF	USERDIDFILTER	RACF resource access list	`uid=CTGuser1,ou=TMS, dc=CTGTest,o=CTG`
RACF	REGISTRY	RACF	`ctg-test-registry.company.com:389`

Steps in this scenario

Prerequisites

You must ensure that the prerequisites for CICS Transaction Server, CICS Transaction Gateway, and WebSphere Application Server are satisfied.

Here are the system prerequisites for CICS Transaction Server for z/OS:

- The CICS server must be CICS Transaction Server for z/OS Version 4.1 or later, with the APAR fixes described in "Configurations that support identity propagation" on page 51.
- The z/OS identity propagation function must be that provided by z/OS, Version 1.11 or later.
- 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.
- The TCP/IP network must extend between LPARs if CICS Transaction Gateway for z/OS and the CICS server exist on different LPARs.
- You must set the SEC system initialization parameter to YES to enable security.
- You must have valid RACF user IDs and passwords.

Here are the system prerequisites for CICS Transaction Gateway:
- CICS Transaction Gateway must be installed on the same sysplex as CICS Transaction Server. For more information see "Sysplex restrictions" on page 15.
- The CICS Transaction Gateway STDENV file must be correctly configured. For more information see "STDENV file" on page 98.

Here are the system prerequisites for WebSphere Application Server:
- Administrative security must be enabled in WebSphere Application Server.
- WebSphere Application Server must already be configured to use an LDAP server and LDAP authentication, and a Distinguished Name (DN) must exist.
- The CICS Transaction Gateway ECI resource adapter must be installed on WebSphere Application Server. For more information, see "Deploying the ECI resource adapter on WebSphere Application Server for z/OS" on page 173.

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 ECIDateTime sample EJB application supplied with CICS Transaction Gateway. This sample must be installed onto WebSphere Application Server.
- The sample CICS Transaction Gateway server program EC01 must be compiled, defined, and installed on CICS.

Configuring identity propagation on CICS TS

You must configure the TCPIPService and the IPCONN on CICS TS.

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, IPICSRV. 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; for example, 50889.
   - 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.
2. Install the CEDA definition.

Configuring the IPCONN on CICS TS

You must define the IPCONN for the incoming IPIC connection.

The following parameters are configured in this step:

Parameter	Purpose
APplid	Set this to match the APPLID specified in the ctg.ini file.
Networkid	Set this to match the APPLIDQUALIFIER specified in the ctg.ini file.
TCPIPService	Set this to match the name of the TCPIPService in CICS.
Userauth	Set this to `Identify`.
SENdcount	Set this value to zero; for more information, see “Configuring IPIC on CICS Transaction Server for z/OS” on page 111

1. Use CEDA to define an IPCONN, for example IPICIP, to include the settings shown in the table. Leave all the other parameters including IDprop with their default settings. The IDprop parameter is not applicable to CICS Transaction Gateway to CICS communication; it is used exclusively for CICS to CICS communication.

   The following example shows an IPCONN definition that has been defined using the CEDA transaction:

```
CEDA View Ipconn( IPICIP )
 Ipconn          : IPICIP
 Group           :
 Description     :
IPIC CONNECTION IDENTIFIERS
 APplid          : MYAPPL
 Networkid       : MYQUAL
 Host            :
 (Lower Case)    :
 Port            : No        No | 1-65535
 Tcpipservice    : IPICSRV
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 | Certuser
 SECurityname    :
 Userauth        : Identify   Local | Identify | Verify | Defaultuser
 IDprop          : Optional   Not allowed | Optional | Required
RECOVERY
 Xlnaction       : Keep       Keep | Force
```

2. Install the IPCONN definition and check that the output from the CEMT INQ IPCONN(IPICIP) command identifies it as “Inservice Released” (`Ins Rel Nos`) in the output:

   ```
   CEMT I IPCONN
    STATUS: RESULTS - OVERTYPE TO MODIFY
    Ipc(IPICIP) App(MYAPPL ) Net(MYQUAL ) Ins Rel Nos
    Rece(100) Sen(000) Tcp(IPICSRV)
   ```

Configuring identity propagation on CICS TG

You must edit the ctg.ini file to set values for the APPLID, APPLIDQUALIFIER, and IPICSERVER parameters.

1. Identify the Gateway daemon to CICS for the connection over IPIC. Do this by setting values for the APPLID and APPLIDQUALIFIER parameters. The values must be in uppercase and are set in the PRODUCT section of the ctg.ini file; for example:

   ```
   SECTION PRODUCT
       APPLID=MYAPPL
       APPLIDQUALIFIER=MYQUAL
   ENDSECTION
   ```

2. 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:

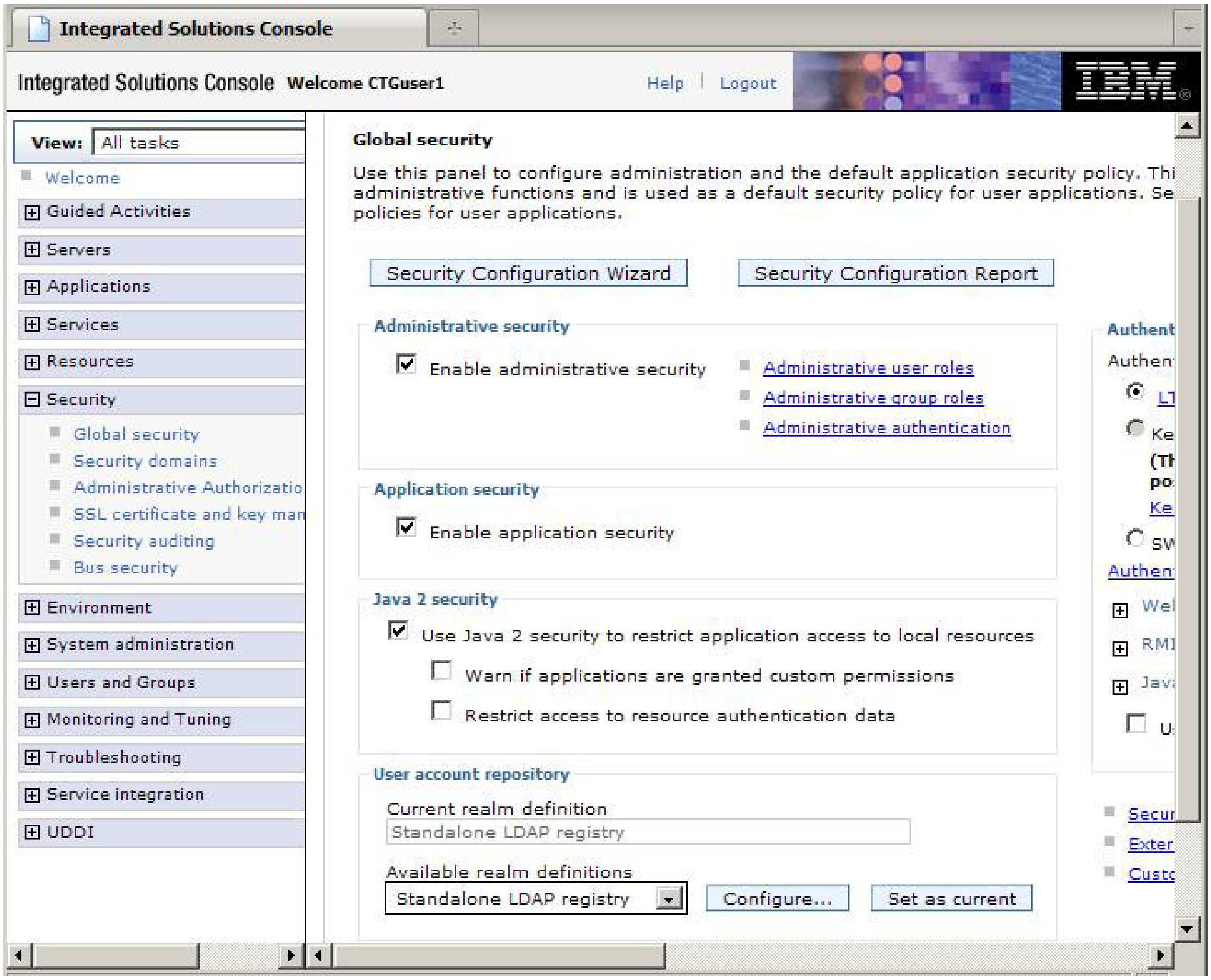
      ```
      SECTION IPICSERVER = CICSA
          HOSTNAME=cicssrv2.company.com
          PORT=50889
      ENDSECTION
      ```

3. Save your updated ctg.ini file.
4. Start CICS Transaction Gateway to apply the new definitions.

Configuring identity propagation on WebSphere Application Server

You must perform the configuration tasks that set up identity propagation on WebSphere Application Server.

1. Enable application security. Use the administrative console; navigate to **Security > Global security** and select **Enable application security**:



Global security

Use this panel to configure administration and the default application security policy. Thi
administrative functions and is used as a default security policy for user applications. Se
policies for user applications.

Administrative security

☑ Enable administrative security

Application security

☑ Enable application security

Java 2 security

☑ Use Java 2 security to restrict application access to local resources

☐ Warn if applications are granted custom permissions

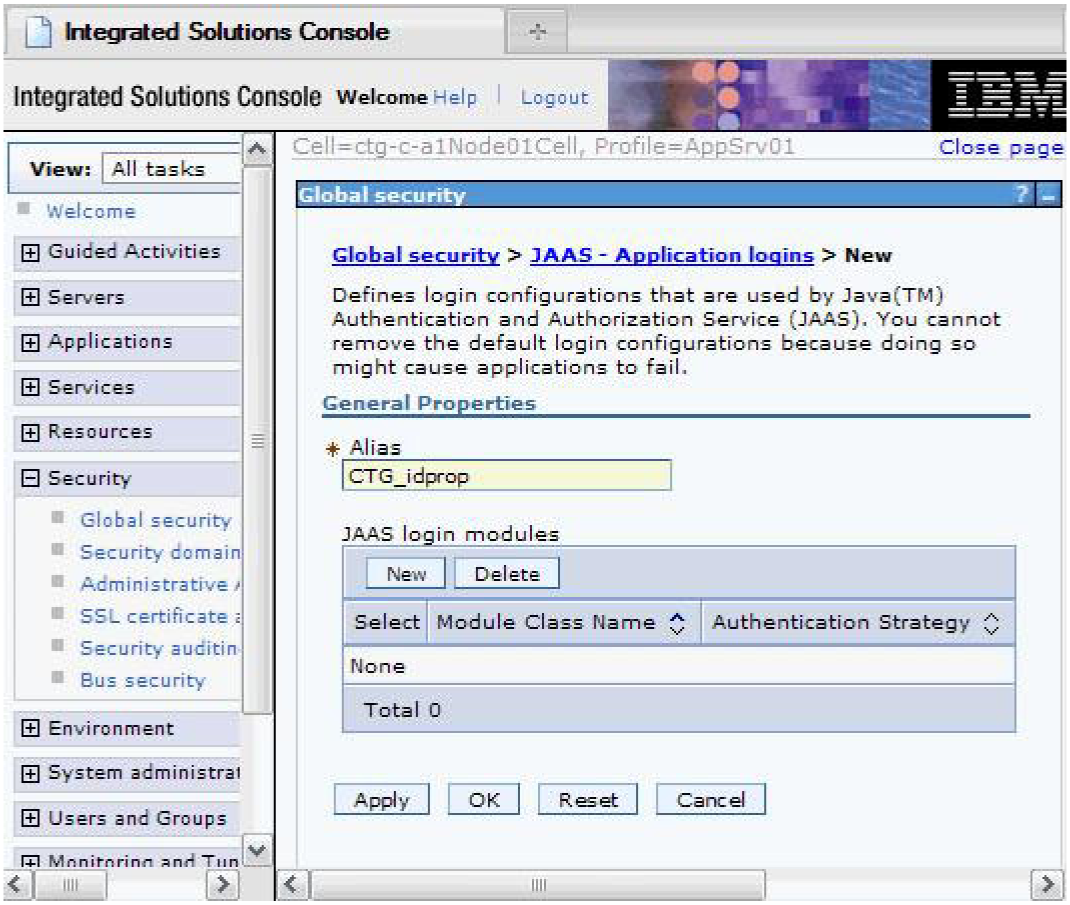
☐ Restrict access to resource authentication data

User account repository

Current realm definition: Standalone LDAP registry

Available realm definitions: Standalone LDAP registry

2. Install the identity propagation login module:
   a. From the WebSphere administrative console; click **Security** > **Global security**, and expand **Java Authentication and Authorization Service**. Click **Application logins** and select **New**. Give this login the alias `CTG_idprop` and click **OK**.



Global security > JAAS - Application logins > New

Defines login configurations that are used by Java(TM) Authentication and Authorization Service (JAAS). You cannot remove the default login configurations because doing so might cause applications to fail.

* Alias: CTG_idprop

JAAS login modules

Select	Module Class Name	Authentication Strategy
None		
Total 0		

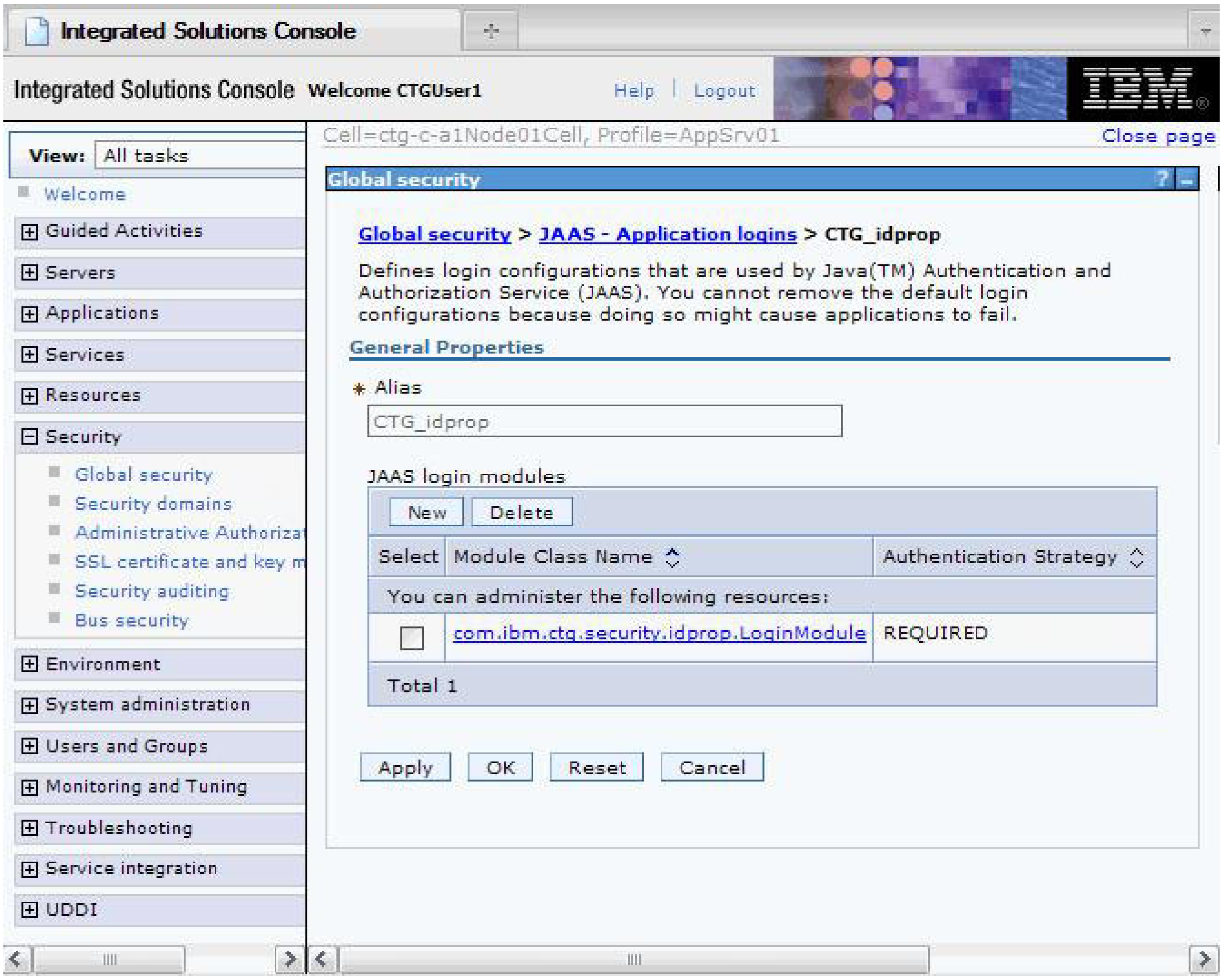
b. From the WebSphere administrative console; click **Security** > **Global security**, and expand **Java Authentication and Authorization Service**. Click **Application logins**, select **CTG_idprop** and click **New**.

c. Set the module class name to `com.ibm.ctg.security.idprop.LoginModule`.

d. Select REQUIRED from the **Authentication strategy** drop down list.

e. Under Custom properties create an entry with the Name set to `propIdentity` and Value set to `RunAs`.

f. Click OK.



Global security > JAAS - Application logins > CTG_idprop

Defines login configurations that are used by Java(TM) Authentication and Authorization Service (JAAS). You cannot remove the default login configurations because doing so might cause applications to fail.

* Alias: CTG_idprop

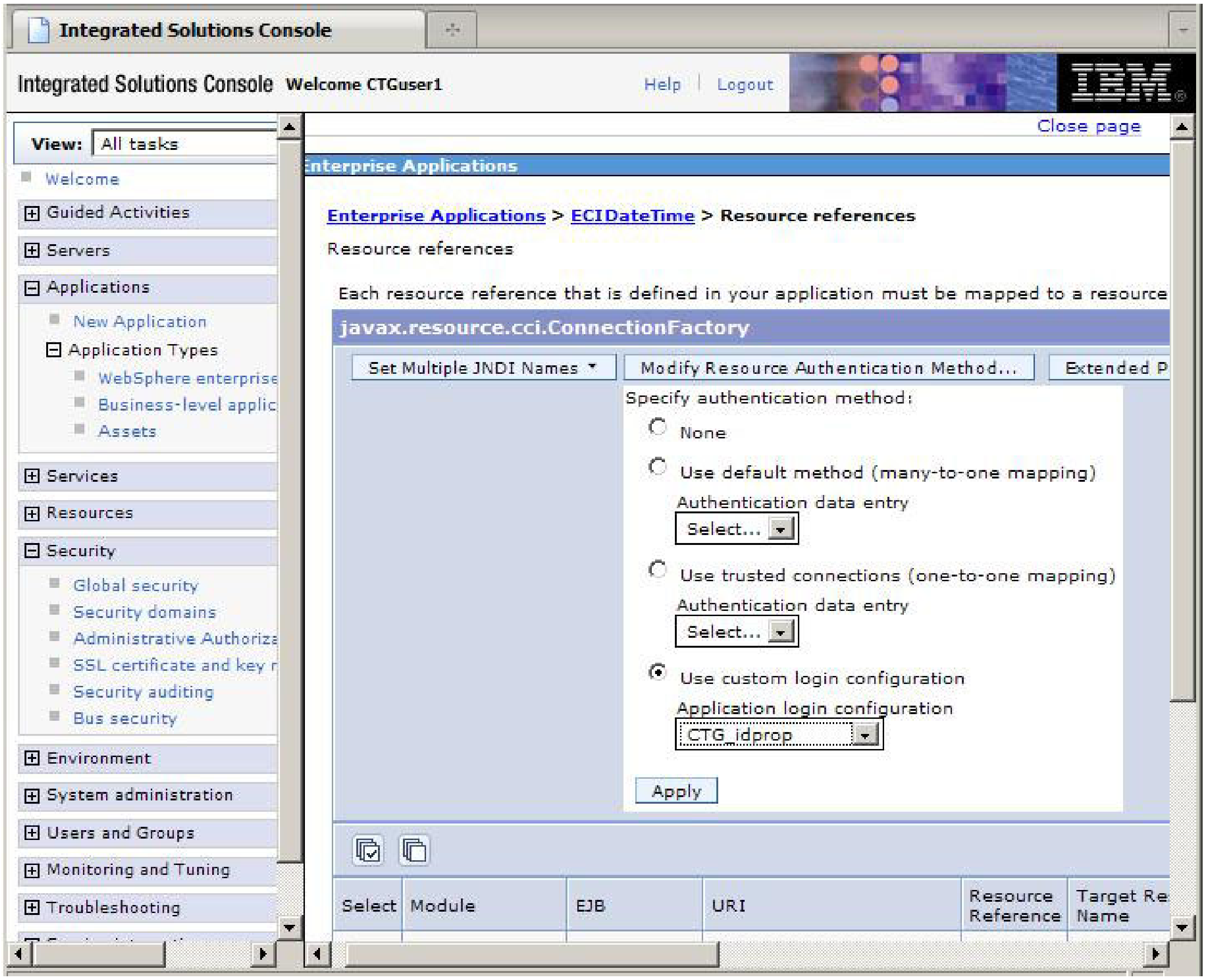
JAAS login modules

You can administer the following resources:

Select	Module Class Name	Authentication Strategy
☐	com.ibm.ctg.security.idprop.LoginModule	REQUIRED

Total 1

3. Associate the identity propagation login module with the client application:
   a. From the administrative console, navigate **Applications > Application types > WebSphere enterprise applications** and select the ECIDateTime application from the list.
   b. Select **Resource references**.
   c. Select the ECI resource reference using the checkbox, and then select **Modify resource authentication method**.
   d. Select **Use Custom login configuration** and then select the identity propagation login module `CTG_idprop` that you installed in the previous step.
   e. Click **Apply**, to apply the selected identity propagation login module to the ECI resource reference
   f. Click **OK**



Enterprise Applications > ECIDateTime > Resource references

Resource references

Each resource reference that is defined in your application must be mapped to a resource

javax.resource.cci.ConnectionFactory

Specify authentication method:

○ None

○ Use default method (many-to-one mapping)

Authentication data entry: Select...

○ Use trusted connections (one-to-one mapping)

Authentication data entry: Select...

● Use custom login configuration

Application login configuration: CTG_idprop

4. Save the configuration changes that you have made so far.

Checking that the connection is secure

To check that the connection is secure you run the ECIDateTime application.

LDAP names and RACF names have not yet been mapped using RACMAP. As a result, the application returns various error messages. The appearance of these messages at this stage is to be expected and is normal. If some messages *do not* appear, their nonappearance might indicate a problem such as security setting SEC=NO, a default user ID that has too much authority, or RACF mapping that has already been defined.

1. To run the ECIDateTime application, start the launchClient utility from the command line by issuing the following command:

   ```
   app_server_root/bin/launchClient filepath/ECIDateTime.ear
   ```

   Where *filepath* is the path to the ECIDateTime .ear file.
2. With application security enabled, when you run the ear file from launchClient, a dialog prompts you to supply the security credentials (username and password). Because the application is being authenticated against the LDAP registry, you must supply the "username" (in reality a Distinguished Name) that has been defined in the LDAP registry

(uid=CTGuser1,ou=TMS,dc=CTGTest,o=COMPANYCTG). The password is also required. You now see the return code ECI_ERR_SECURITY_ERROR and a Java stack trace in the console. The exception starts as follows:

```
javax.resource.spi.SecurityException: CTG9631E Error occurred during
interaction with CICS: ECI_ERR_SECURITY_ERROR, error code: -27
```

3. Check the CICS user message log and the JES message log for the CICS job. A message confirms that the error occurred because identity propagation has not yet been configured. The JES message log for the CICS job contains this message:

   ```
   11.36.45 JOB09604 ICH408I USER(TESTID ) GROUP(TSOUSER ) NAME(TEST )
   113 113 DISTRIBUTED IDENTITY IS NOT DEFINED:
   113 uid=CTGuser1,ou=TMS,dc=CTGTest,o=COMPANYCTG
           ctg-test-registry.ibm.com:389
   11.36.45 JOB09604 IRR012I VERIFICATION FAILED. USER PROFILE NOT FOUND
   ```

   The CICS user message log contains this message:

   ```
   DFHIS1027 10/26/2009 11:36:45 IY24CTGC Security violation has been detected
   using IPCONN IPICIP and transaction id CSMI by userid BADLINK
   ```

   Note: In this example, BADLINK is the CICS default user ID defined in the DFLTUSER system initialization parameter, and does not have permission to run the CSMI transaction.

If these messages appear, this is not an indication of a problem. On the contrary, the messages are expected because although the connection to CICS was established, the application failed because the LDAP identity was not propagated through to CICS. In the next step “Configuring identity propagation on RACF,” you configure the mapping between LDAP and RACF identities.

Configuring identity propagation on RACF

You must complete the required configuration tasks on RACF.

To map your distributed identity to your RACF user ID issue the following command:

```
RACMAP ID(TESTID) MAP
USERDIDFILTER(NAME('uid=CTGuser1,ou=TMS,dc=CTGTest,o=COMPANYCTG'))
REGISTRY(NAME('ctg-test-registry.ibm.com:389'))
```

For more information about the RACMAP command, see the*z/OS Security Server RACF Command Language Reference*.

Testing your scenario

If you have configured your identity mapping correctly, you can now run the ECIDateTime application successfully to test the scenario and verify that the application is being run with the correctly mapped RACF ID.

1. On CICS, start CEDX debugging on the CSMI mirror transaction on the CICS server by issuing the following CEDX CSMI command:

   ```
   CEDX CSMI
   ```

   This switches on the debug tool.
2. On WebSphere Application Server, run the ECIDateTime application. To do this, start launchClient from the command line by issuing the following command:

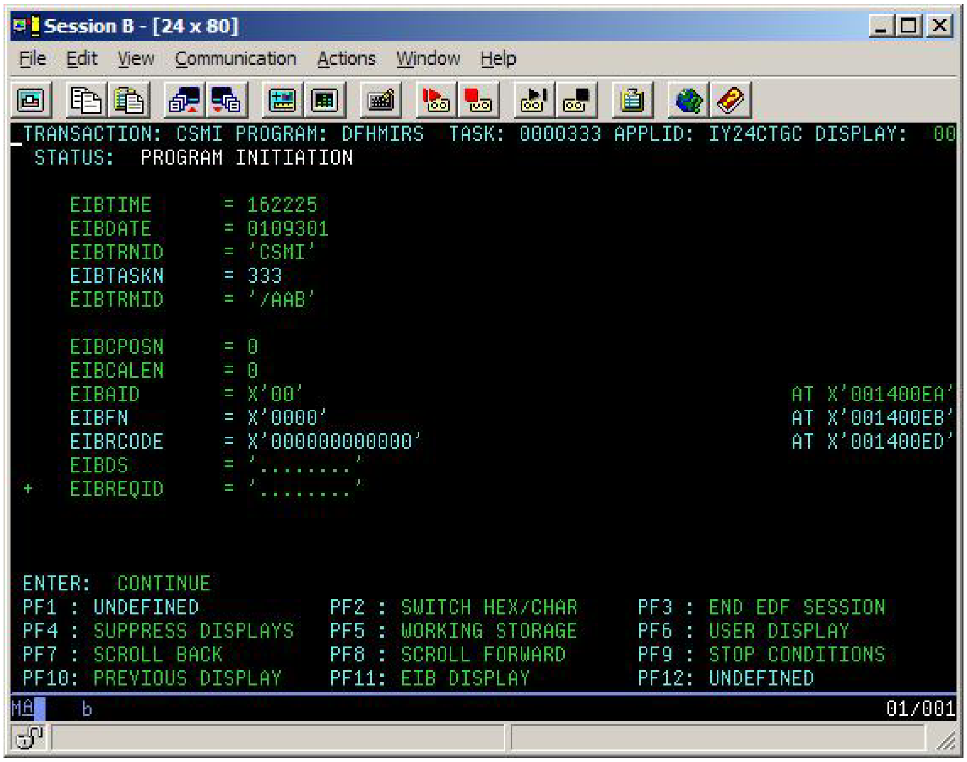
   ```
   app_server_root/bin/launchClient path/ECIDateTime.ear
   ```

   Where *path* is the path to the ECIDateTime.ear file.

3. On WebSphere Application Server, when prompted for a user name and password, provide the user name and password defined in the LDAP server:

   `uid=CTGuser1,ou=TMS,dc=CTGTest,o=COMPANYCTG`

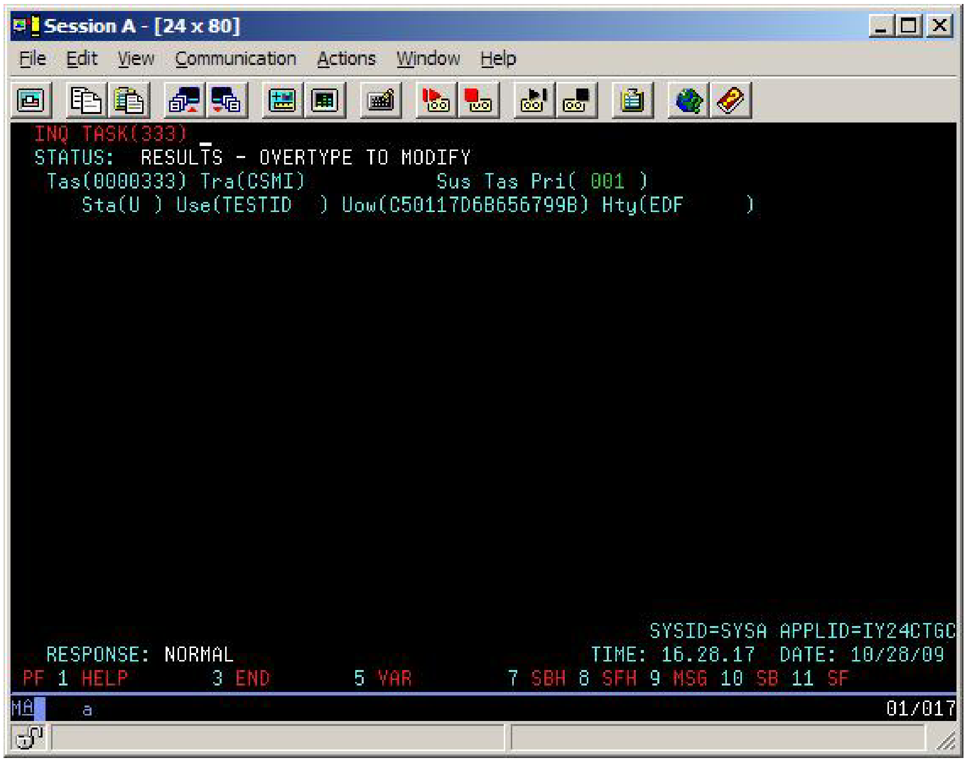
   On CICS, when the transaction starts, CEDX intercepts the program and displays the program initiation screen until you press the Enter key. You can now examine the task ID number before the transaction completes:



```
TRANSACTION: CSMI PROGRAM: DFHMIRS  TASK: 0000333 APPLID: IY24CTGC DISPLAY:  00
 STATUS:  PROGRAM INITIATION

    EIBTIME      = 162225
    EIBDATE      = 0109301
    EIBTRNID     = 'CSMI'
    EIBTASKN     = 333
    EIBTRMID     = '/AAB'

    EIBCPOSN     = 0
    EIBCALEN     = 0
    EIBAID       = X'00'                                        AT X'001400EA'
    EIBFN        = X'0000'                                      AT X'001400EB'
    EIBRCODE     = X'000000000000'                              AT X'001400ED'
    EIBDS        = '........'
+   EIBREQID     = '........'



ENTER:  CONTINUE
PF1 : UNDEFINED            PF2 : SWITCH HEX/CHAR      PF3 : END EDF SESSION
PF4 : SUPPRESS DISPLAYS    PF5 : WORKING STORAGE      PF6 : USER DISPLAY
PF7 : SCROLL BACK          PF8 : SCROLL FORWARD       PF9 : STOP CONDITIONS
PF10: PREVIOUS DISPLAY     PF11: EIB DISPLAY          PF12: UNDEFINED
MA     b                                                                01/001
```

Figure 36. Program initiation screen

4. On CICS, use a second sign-on to log on to the server and run CEMT INQ TASK on the CSMI task:



```
 INQ TASK(333)
 STATUS:  RESULTS - OVERTYPE TO MODIFY
  Tas(0000333) Tra(CSMI)           Sus Tas Pri( 001 )
     Sta(U ) Use(TESTID  ) Uow(C50117D6B656799B) Hty(EDF     )




















                                                 SYSID=SYSA APPLID=IY24CTGC
  RESPONSE: NORMAL                             TIME: 16.28.17  DATE: 10/28/09
PF 1 HELP       3 END       5 VAR        7 SBH 8 SFH 9 MSG 10 SB 11 SF
MA    a                                                                 01/017
```

Figure 37. CEMT INQ task screen

The `Use` field contains the RACF ID to which the distinguished name maps. The RACF ID in this scenario is `TESTID` and is evidence that the scenario is operating correctly.

5. On CICS, unlock the CEDX session by pressing the PF3 key. On WebSphere Application Server, the launchClient returns the date and time:

```
WSCL0014I: Invoking the Application Client class
com.ibm.ctg.samples.jee.ECIDateTimeClient
CICS Date/Time=27/10/09 15:48:45
```

Configuring SSL security between a Java Client and the Gateway daemon (SC05)

This scenario shows you how to 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.

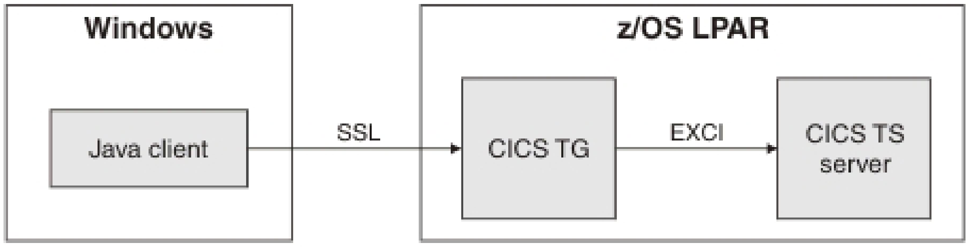


Figure 38. Topology used in this scenario

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	port	In the protocol@ssl.parameters parameters in the SECTION GATEWAY in ctg.ini	8573
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	CTGKEYRING
CICS TG	esmkeyring	SECTION PRODUCT in ctg.ini	on
RACF	user ID	RACDCERT command	CTGUSER
RACF	name (self-signed certificate)	RACDCERT command	CTG CA CERT
RACF	name (personal certificate)	RACDCERT command	CTG PERSONAL CERT
RACF	name (keyring)	SECTION PRODUCT in ctg.ini	CTGKEYRING
RACF	filename (personal certificate)	RACDCERT command	CTGUSER.PERSONAL.CERT
Java Client	filename (personal certificate)	FTP command	client.personal.cert.arm
Java Client	keyring filename	iKeyman	myclientkeyring.jks
Java Client	password	iKeyman	mypassword
Java Client	label	iKeyman	cics tg racf server certificate

Prerequisites for the SSL scenario

The prerequisites for this SSL scenario.

Here are the system requirements for CICS Transaction Server for 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.

- The TCP/IP network must extend between LPARs if CICS Transaction Gateway for z/OS and the CICS server exist on different LPARs.
- You must have valid RACF user IDs and passwords.

Here are the system requirements for CICS Transaction Gateway:
- CICS Transaction Gateway must be installed.
- A working connection from CICS Transaction Gateway to CICS is required. This can be IPIC or EXCI. This scenario uses an EXCI connection. For more information see "Configuring EXCI" on page 119.

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 Transaction Server program EC01 to be compiled, defined, and installed on CICS.
- The CICS Transaction Gateway supplied Java sample EciB2 be available on the Java client machine.

Configure SSL server authentication - step 1

This step involves using RACF commands to create a CA certificate, a signed personal certificate and a keyring on the server.

You perform these tasks on the z/OS platform by issuing RACDCERT (RACF digital certificate) commands. The RACDCERT commands enable you to create and maintain digital certificates, and to create the keyrings which act as repositories for digital certificates.

1. Create a CA certificate that is self signed on the server (in RACF):
   ```
   RACDCERT CERTAUTH GENCERT SUBJECTSDN(OU('CTG TEST') O('IBM')
   T('CTG CA CERT') C('GB')) KEYUSAGE(CERTSIGN) WITHLABEL('CTG CA CERT')
   ```
2. Refresh the RACF class:
   ```
   SETR RACLIST(DIGTCERT) REFRESH
   ```
3. Check that the CA certificate has been created; do this by verifying that it exists in the output from listing the DIGTCERT class:
   a. Open ISPF.
   b. From the ISPF main menu select **R RACF**.
   c. From the **RACF - SERVICES OPTION MENU** screen select **2 GENERAL RESOURCE PROFILES**.
   d. From the **GENERAL RESOURCE PROFILE SERVICES** screen select **D or 8 DISPLAY PROFILE CONTENTS**.
   e. From the **DISPLAY GENERAL RESOURCE PROFILE** screen do the following:

      Enter the class name DIGTCERT in the **CLASS** field.

      Leave the **PROFILE** field blank.

      Enter YES to select the profile type **DISCRETE**.

      Enter YES to select the **ACCESS LIST** option.

   Press **Enter** to display a list of the selected classes and confirm that it contains the DIGTCERT class you have just created.
4. List the certificate:
   ```
   RACDCERT CERTAUTH LIST(LABEL('CTG CA CERT'))
   ```

5. Generate a personal certificate on the server and sign it with your CA certificate:

   ```
   RACDCERT ID(CTGUSER) GENCERT SUBJECTSDN(OU('CTG TEST') O('IBM')
   T('CTG PERSONAL CERT') C('GB')) WITHLABEL('CTG PERSONAL CERT')
   SIGNWITH(CERTAUTH LABEL('CTG CA CERT'))
   ```

   Where CTGUSER is a valid RACF user ID.
6. Refresh the RACF class:

   ```
   SETR RACLIST(DIGTCERT) REFRESH
   ```
7. Create a keyring where certificates are stored:

   ```
   RACDCERT ADDRING(CTGKEYRING) 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(CTGKEYRING) USAGE(CERTAUTH))
      ```
   b. Add the personal certificate to the keyring:

      ```
      RACDCERT ID(CTGUSER) CONNECT(LABEL('CTG PERSONAL CERT')
      RING(CTGKEYRING)
      DEFAULT USAGE(PERSONAL))
      ```
9. List the keyring to confirm that it contains the certificates:

   ```
   RACDCERT LISTRING(CTGKEYRING) ID(CTGUSER)
   ```

   Here is an example of the output generated by this command:

   ```
   Ring:
        >CTGKEYRING<
   Certificate Label Name             Cert Owner    USAGE       DEFAULT
   --------------------------------   -----------   -----       -------
   CTG CA CERT                        CERTAUTH      CERTAUTH    NO
   CTG PERSONAL CERT                  ID(CTGUSER)   PERSONAL    YES
   ```
10. Export the personal certificate to a file on the server:

    ```
    RACDCERT ID(CTGUSER) EXPORT(LABEL('CTG PERSONAL CERT'))
    DSN('CTGUSER.PERSONAL.CERT') FORMAT(CERTB64)
    ```

    The FORMAT(CERTB64) specifies that the certificate is stored in ASCII format. Use ISPF 3.4 to view the certificate.

Configure SSL server authentication - step 2

This step involves using FTP to transfer the signed personal certificate from the server to the client machine, then using ikeyman to create a Java keystore (jks) file where the certificate is then stored.

ikeyman is provided as part of the Java Runtime Environment.

1. Transfer the personal certificate to your Client machine using either an FTP client or the command line.

   In the previous task, 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, for example:

   ```
   C:\CICSTG>ftp server
   Connected to server.company.com
   User (server.company.com:(none)): name
   331 Send password please. Password: 230 name is logged on.
   Working directory is "/u/directory".
   ftp> asc
   200 Representation type is Ascii NonPrint
   ftp> get 'CTGUSER.PERSONAL.CERT'
   ```

```
200 Port request OK. 125 Sending data set CTGUSER.PERSONAL.CERT 250
Transfer completed successfully.
ftp: 976 bytes received in 0.02Seconds 61.00Kbytes/sec.
ftp> quit
Rename 'CTGUSER.PERSONAL.CERT' to client.personal.cert.arm
```

2. Create a Java keystore file on your Client machine using ikeyman. To start ikeyman, go to the directory containing ikeyman and double-click the ikeyman exe file for example:

   ```
   C:\Program Files\IBM\Java60\jre\bin\ikeyman.exe
   ```

   a. From the ikeyman main menu select **Key Database File** > **New**.
   b. From the New dialog, click the **Key database type** list then select 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 myclientkeyring.jks)
   d. Click **OK** to confirm.
   e. 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).
   f. Click **OK** to confirm.
3. Import the personal certificate from the file into the Java keystore file.
   a. Click the downward arrow and select **Signer certificates** from the list.
   b. Click **Add** and specify the filename and location of the file that you transferred to the client. This imports the server personal certificate from the file into the Java keystore file.
   c. Click **OK**.
   d. Enter a label for the certificate in the **Enter a label** dialog. The label provides a way of identifying the certificate but is not used during security processing. This scenario used “cics tg racf server certificate”.
   e. Click **OK**. This imports the server personal certificate from the file that you transferred to the client, into the Java keystore file.

Configure SSL client authentication (optional)

SSL client authentication can optionally be configured if you have already configured SSL server authentication

You perform some of these tasks on the z/OS platform by issuing RACDCERT (RACF digital certificate) commands. The RACDCERT commands enable you to create and maintain digital certificates, and to create the keyrings which act as repositories for digital certificates. You also use ikeyman.

ikeyman is provided as part of the Java Runtime Environment.

1. Create a CA certificate on your Client that is self signed. Start ikeyman and open the Java keystore (.jks) file.
2. Click **Create** > **New Self-Signed Certificate**. In the Create New Self-Signed Certificate window, complete the following steps:
   a. In the **Key label** field, type cics tg client certificate. This provides a way of identifying the certificate, and is not used in security processing.
   b. On the **Version** menu, select X509 V3.
   c. On the **Key size** menu, select 1024.

   d. The Common name defaults to the name of the machine you are using, and the Validity period defaults to 365 days.
   e. Click **OK**.
3. Select the new personal certificate by selecting Personal Certificates on the dropdown menu that currently displays Signer Certificates and click **Extract Certificate...**. Specify the data type “Base64-encoded ASCII data”, the certificate file name (this scenario used client.personal.cert.arm), and the location (for example C:\CICSTG).
4. Check to ensure that the file is visible in the folder.
5. Transfer the file to the server, using either FTP or the command line, for example:

   ```
   C:\CICSTG>ftp server
   Connected to server.company.com
   User : ctguser
   Password : xxx
   CTGUSER is logged on. Working directory is "/u/ctguser".
   ftp> asc
   Representation type is Ascii NonPrint
   ftp> literal site recfm=vb
   200 SITE command was accepted
   ftp> cd 'CTGUSER'
   "CTGUSER." is the working directory name prefix.
   ftp> put client.personal.cert.arm
   Storing data set CTGUSER.CLIENT.PERSONAL.CERT.ARM
   ftp> quit
   ```
6. Add to the certificate to RACF:

   ```
   RACDCERT ID (CTGUSER) ADD('CTGUSER.CLIENT.PERSONAL.CERT.ARM') WITHLABEL
   ('MY CLIENT CERT') TRUST
   ```

   The following message is displayed:

   ```
   The new profile for DIGTCERT will not be in effect until a SETROPTS
   REFRESH has been issued.
   Certificate Authority not defined to RACF. Certificate added with
   TRUST status.
   ```
7. Refresh the RACF repository on the server:

   ```
   setr raclist(digtcert) refresh
   ```
8. Add the client certificate to the keyring:

   ```
   RACDCERT ID (CTGUSER) CONNECT(LABEL ('MY CLIENT CERT')
   RING(CTGKEYRING) USAGE (CERTAUTH))
   ```
9. Check that the server personal certificate has been added to the keyring:
   a. Open ISPF.
   b. From the ISPF main menu select **R RACF**.
   c. From the **RACF - SERVICES OPTION** menu select **7 DIGITAL CERTIFICATES, KEYRINGS, AND TOKENS**.
   d. From the **DIGITAL CERTIFICATES AND RELATED FUNCTIONS** menu select **2 KEYRING FUNCTIONS**.
   e. From the **DIGITAL CERTIFICATE KEYRING SERVICES** menu, specify user CTGUSER and select the option **3 LIST EXISTING KEYRINGS**.
   f. At the prompt “Enter specific ring names or an asterisk * to list up to 4 rings”, enter an asterisk (*).

   ISPF now lists the available certificate label names:

   ```
   Ring:
        >CTGKEYRING<
   Certificate Label Name               Cert Owner    USAGE       DEFAULT
   ```

```
--------------------------------   -----------   -----      -------
CTG CA CERT                        CERTAUTH      CERTAUTH    NO
CTG PERSONAL CERT                  ID(CTGUSER)   PERSONAL    YES
MY CLIENT CERT                     ID(CTGUSER)   CERTAUTH    NO
```

Configuring the Gateway daemon for SSL

Updating the CICS Transaction Gateway configuration file (ctg.ini) for SSL.

1. Edit the ctg.ini file to add the SSL handler definition `protocol@ssl.handler=com.ibm.ctg.server.SslHandler`
2. Add the following parameters to the SSL handler parameters definition `protocol@ssl.parameters`:

   port This parameter identifies the TCP/IP port on which the protocol handler listens for incoming client requests.

   clientauth
   This parameter determines whether or not client authentication occurs. Valid values are on, client authentication is performed, or off, client authentication is not performed. The default is off.

   For example, if you have taken the optional step of configuring the SSL client authentication:

   ```
   protocol@ssl.parameters=port=8573;\
                           clientauth=on;
   ```

3. Add the following parameters to the PRODUCT section:

   keyring
   This parameter specifies the name of the keyring to be used by this protocol handler. For more information, see “Key ring file” on page 138

   esmkeyring
   This parameter specifies that the SSL keyring is stored in an external security manager (ESM). For more information, see “ESM key ring” on page 139

   For example:

   ```
   SECTION PRODUCT
   KEYRING=CTGKEYRING
   ESMKEYRING=ON
   ENDSECTION
   ```

4. Save the changes.

Verifying that SSL is enabled on the connection

Verify that SSL security is enabled on the Java client connection to CICS Transaction Gateway.

Start CICS Transaction Gateway. 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 that is 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 the following message:

```
CTG6525E Unable to start handler for the ssl: protocol, port: 8573, because:
```

The message is followed by a reason, for example `invalid port number` and a Java exception. If an exception occurs, rectify the problem and restart CICS Transaction Gateway.

Testing the SSL scenario

Set the environment variables then test the scenario by starting CICS Transaction Gateway and sending an ECI request to the CICS server.

Set the environment variables

If you have not already done so, you must set the Java CLASSPATH environment variable so that the ctgclient.jar and ctgsamples.jar files can be found.

To set the CLASSPATH environment variable:

1. Open a command prompt window and go to the directory where the Java keystore file is located.
2. Set the Java CLASSPATH environment variable by issuing the set CLASSPATH command, for example:

   ```
   set CLASSPATH=<install_path>\classes\ctgclient.jar;<install_path>\classes\
      ctgsamples.jar;%CLASSPATH%
   ```

Send an ECI request to CICS

When you send an ECI request from the Java client application EciB1 specifying the ssl:// protocol, an SSL handshake between the Java client and the Gateway daemon is attempted. When server authentication, and if configured, client authentication have been successful, the Gateway daemon lists the available CICS servers to forward the ECI request to. When you have selected your CICS server, the CICS application EC01 returns the current date and time.

The source for the EciB1 application is located in the samples folder, for example:

```
<install_path>\samples\java\com\ibm\ctg\samples\eci
```

To test the scenario:

1. Start CICS Transaction Gateway.
2. Send the ECI request to the CICS server by issuing a Java command that has 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://server.company.com 8573
myclientkeyring.jks MyPassword
```

3. The command returns a list of CICS servers, for example:

```
CICS Servers Defined:

        1. CICSA -CICS V4.1 Server

Choose Server to connect to, or q to quit:
```

4. When prompted, type the number of the CICS server to which you want the ECI request sent.

The CICS server returns the current date and time, for example:

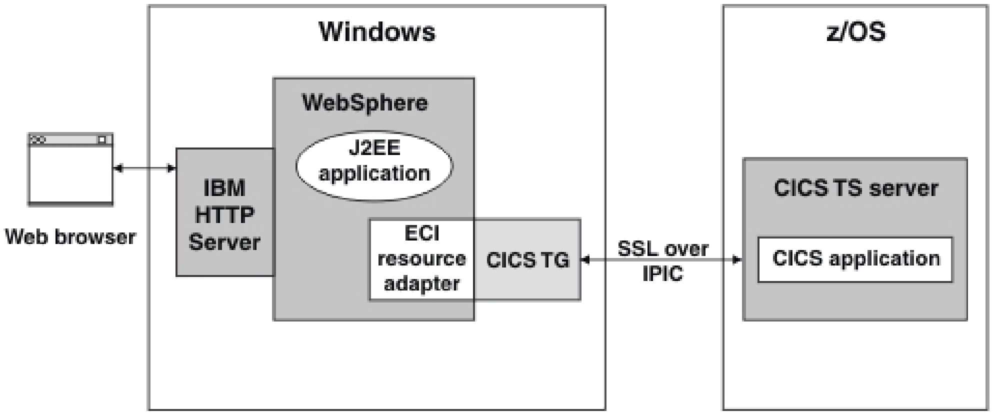
```
Program EC01 returned with data:-

        Hex: 32382f30312f31302031353a33323a34360
        ASCII text: 28/01/10 15:32:46
```

You have now successfully completed the scenario.

Configuring SSL between CICS TG and CICS (SC07)

This scenario shows you how to configure SSL security on an IPIC connection between CICS Transaction Gateway running in local mode on WebSphere Application Server V8.0 and CICS Transaction Server V4.1.

The following figure shows the topology used in this scenario:



Follow the step-by-step instructions in this scenario using these values:

Component	Parameter	Where set	Example value
CICS server	user ID		CTGUSER
CICS server	CA certificate name	RACDCERT command	CTG CA CERT
CICS server	personal certificate name	RACDCERT command	CTG PERSONAL CERT
CICS server	keyring name	RACDCERT command	CICSSERVERKEYRING

Component	Parameter	Where set	Example value
CICS server	personal certificate file name	RACDCERT command	CTGUSER.PERSONAL.CERT
CICS server	TCPIPService	TCPIPService definition	SSL51190
CICS server	port	TCPIPService definition	51190
Java client	personal certificate file name	ikeyman	personalcert.arm
Java client	keyring file name	ikeyman	ctgclientkeyring.jks
Java client	keyring password	ikeyman	MyPassword
Java client	CTG_APPLID	WebSphere Application Server	SSLAH

Prerequisites for the SSL scenario

Before you can complete this scenario, you must ensure that the system requirements for CICS Transaction Server, CICS Transaction Gateway, and WebSphere Application Server are satisfied.

CICS Transaction Server:

- The CICS server version must be CICS Transaction Server V3.2 or later because IPIC is not available in earlier releases of CICS.
- TCP/IP services must be active in the CICS server.
  - To activate these services, set the 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.

WebSphere Application Server:

- WebSphere Application Server must be installed on the same machine as CICS Transaction Gateway.

To test that the scenario works successfully you can use either the supplied samples, or your own applications. If you use the supplied samples you must complete the following tasks:

- Install the sample CICS COBOL programs EC01, EC03 on the CICS server.

For information about the samples see

Configuring SSL server authentication on the CICS server

To complete this task you use RACF commands to create a CA certificate, a signed personal certificate, and a keyring on the CICS server.

You perform this task by issuing ISPF RACDCERT (RACF digital certificate) commands. You use RACDCERT commands to create and maintain digital certificates, and create the keyrings that are the repositories for digital certificates.

1. Create a CA certificate that is self-signed on the server (in RACF):

```
RACDCERT CERTAUTH GENCERT SUBJECTSDN(OU('CTG TEST') O('IBM')
T('CTG CA CERT') C('GB')) KEYUSAGE(CERTSIGN) WITHLABEL('CTG CA CERT')
```

2. Refresh the RACF class:

   ```
   SETR RACLIST(DIGTCERT) REFRESH
   ```

3. Check that the CA certificate has been created by verifying that it exists in the output from listing the DIGTCERT class:
   a. From the ISPF main menu, enter R to display the RACF dialog.
   b. Press Enter.
   c. From the RACF - SERVICES OPTION MENU panel, enter 2 to display the RACF - GENERAL RESOURCE PROFILES panel. Press **Enter**.
   d. From the RACF - GENERAL RESOURCE PROFILE SERVICES panel, enter 8 to display the profile contents. **Press Enter**.
   e. From the RACF - GENERAL RESOURCE SERVICES - DISPLAY panel, type the class name `DIGTCERT` into the **CLASS** field, leaving the **Profile** field blank. Press **Enter**.
   f. From the next RACF - GENERAL RESOURCE SERVICES - DISPLAY panel, complete the following steps:
      1) Ensure that the **CLASS** field contains the class name DIGTCERT.
      2) Leave the **PROFILE** field blank.
      3) In the **DISCRETE** field, enter `Yes`, to select the profile type.
      4) In the **ACCESS LIST** field, enter `Yes` to select the access list option.
      5) Press **Enter**.

   RACF now displays a list of the selected classes; check that the list contains the DIGTCERT class that you have just created.

4. List the certificate:

   ```
   RACDCERT CERTAUTH LIST(LABEL('CTG CA CERT'))
   ```

5. Create a personal certificate on the server and sign it with your CA certificate:

   ```
   RACDCERT ID(CTGUSER) GENCERT SUBJECTSDN(OU('CTG TEST') O('IBM')
   T('CTG PERSONAL CERT') C('GB')) WITHLABEL('CTG PERSONAL CERT')
   SIGNWITH(CERTAUTH LABEL('CTG CA CERT'))
   ```

   CTGUSER must be a valid RACF user ID.

6. Refresh the RACF class:

   ```
   SETR RACLIST(DIGTCERT) REFRESH
   ```

7. Create a keyring where certificates are stored:

   ```
   RACDCERT ADDRING(CTGSERVERKEYRING) ID(CTGUSER)
   ```

8. Add the CA certificate and personal certificate to the keyring:
   a. Add the CA certificate to the keyring:

      ```
      RACDCERT ID(CTGUSER) CONNECT(CERTAUTH LABEL('CTG CA CERT')
      RING(CTGSERVERKEYRING) USAGE(CERTAUTH))
      ```

   b. Add the personal certificate to the keyring:

      ```
      RACDCERT ID(CTGUSER) CONNECT(LABEL('CTG PERSONAL CERT')
      RING(CTGSERVERKEYRING)
      DEFAULT USAGE(PERSONAL))
      ```

9. List the keyring to confirm that it contains the certificates:

   ```
   RACDCERT LISTRING(CTGSERVERKEYRING) ID(CTGUSER)
   ```

   Here is an example of the output generated by this command:

```
Ring:
     >CTGSERVERKEYRING<
Certificate Label Name              Cert Owner    USAGE       DEFAULT
--------------------------------    -----------   -----       -------
CTG CA CERT                         CERTAUTH      CERTAUTH    NO
CTG PERSONAL CERT                   ID(CTGUSER)   PERSONAL    YES
```

10. Export the personal certificate to a file on the server:

    ```
    RACDCERT ID(CTGUSER) EXPORT(LABEL('CTG PERSONAL CERT'))
    DSN('CTGUSER.PERSONAL.CERT') FORMAT(CERTB64)
    ```

    FORMAT(CERTB64) specifies that the certificate is stored in ASCII format.
11. Use ISPF 3.4 to view the certificate.

You have now configured SSL server authentication on the CICS server.

Configuring SSL server authentication on the client

To complete this task you use FTP to transfer the signed personal certificate from the CICS server to the client machine, then iKeyman to create a Java keystore (jks) file where the certificate is stored.

iKeyman is installed in:

- <install_path>\jvm17\bin on Windows
- <install_path>/jvm17/bin on UNIX and Linux

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 227, you specified FORMAT(CERTB64) to ensure that the certificate was stored in ASCII. You must therefore specify ASCII when you transfer the certificate using FTP. The following example shows the FTP commands required to transfer the certificate, and the associated system responses:

   ```
   C:\ftp server
         Connected to server.company.com
         User (server.company.com:(none)): name
         331 Send password please. Password: xxx name is logged on.
         Working directory is "/u/directory".
         ftp> asc
         Representation type is Ascii NonPrint
         ftp> quote site recfm=vb
         SITE command was accepted
         ftp> get 'CTGUSER.PERSONAL.CERT'
         Port request OK. 125 Sending data set CTGUSER.PERSONAL.CERT
         Transfer completed successfully.
         ftp> quit
   ```

   You have to specify the site `recfm=vb` FTP command because the server certificate is stored in a variable blocked data set.
2. Rename CTGUSER.PERSONAL.CERT to personalcert.arm.
3. Start ikeyman on your Client machine.
4. Create a new Java keystore file:
   a. From the iKeyman main menu, select **Key Database File** > **New**.
   b. From the New dialog, click the **Key database type** list then select the file type **JKS**.
   c. In the **File name** field enter the name of the Java keystore file that you want to create. In this scenario the file name is `ctgclientkeyring.jks`.

   d. Click **OK**. Because you are creating a new Java keystore file, the **Password prompt** dialog now prompts you to provide a password. Enter a password into the **Password** and **Confirm password** fields. In this scenario the password is `MyPassword`.
   e. Click **OK**.
5. Import the personal certificate personalcert.arm from the data set into the Java keystore file:
   a. Click the arrow and select **Signer certificates** from the list.
   b. Click **Add** and specify the file name and location of the file that you transferred to the client (in this scenario personalcert.arm).
   c. Click **OK**.
   d. In the **Enter a label** dialog, enter a label for the certificate. The label identifies the certificate but is not used during security processing. This scenario uses the label `cics tg racf server certificate`.
   e. Click **OK**. The server personal certificate is imported from the data set that you transferred to the client, into the Java keystore file.

You have now configured SSL server authentication on the client.

Configuring SSL client authentication

To complete this task you use iKeyman to create and export the client certificate, FTP to transfer the certificate file to the server, and a RACDCERT (RACF digital certificate) command to import the certificate into the RACF keyring.

iKeyman is installed in:
- <install_path>\jvm17\bin on Windows
- <install_path>/jvm17/bin on UNIX and Linux

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 229.
   b. From the menu, select **Personal Certificates**.
   c. Click **New Self-Signed**.
   d. Complete the following mandatory fields:

      Key label
      Enter `exampleclientcert`.

      Version
      Select **X509 V3**.

      Key size
      Select **1024**.

      Common name
      Specify the default value. This is the name of the machine you are using.

Validity period
Specify the default value 365 days.

e. Click **OK**.

The iKeyman tool now generates a public/private key pair.

The self-signed client certificate appears in the Personal Certificates window. The certificate has the name that you entered in the **Key label** field, in this example `exampleclientcert`.

2. Export the client signer certificate:
   a. With `exampleclientcert` highlighted, select **Extract Certificate**.
   b. On the **Data type** menu, select **Base64-encoded ASCII**.
   c. Enter the name and location of the text file containing your Client Certificate data. This scenario uses `exampleclientcert.arm`.
   d. Click **OK**.

   The exported certificate is a signer certificate generated from the personal certificate in the keyring, it does not contain the private key. Import the keyring into the keyring of all servers that need to communicate with the SSL client. The server uses the certificate to verify the identity of the client.
3. Import the client signer certificate into your RACF keyring:
   a. Transfer the file to the server into an MVS sequential data set using FTP, for example:

```
ftp winmvs2g
Connected to server.company.com
User (server.company.com:(none)): name
331 Send password please. Password: xxx name is logged on.
Working directory is "/u/directory".
ftp> asc
Representation type is Ascii NonPrint
ftp> quote site recfm=vb
SITE command was accepted
ftp> put exampleclientcert.arm 'CTGUSER.CLIENT.CERT.ARM'
Port request OK. 125 Sending data set 'CTGUSER.CLIENT.CERT.ARM'
Transfer completed successfully.
ftp> quit
```

   b. Add the client certificate to CLASS(DIGTCERT) using the ISPF RACF command:

```
RACDCERT ID(CTGUSER) ADD('CTGUSER.CLIENT.CERT.ARM')
WITHLABEL('CLIENT.CERT') TRUST
```

   The command returns a message confirming that the certificate has been added with TRUST status and that the class needs to be refreshed:

```
Certificate Authority not defined to RACF. Certificate added with
TRUST status
```

   c. Refresh the RACF class:

```
SETR RACLIST(DIGTCERT) REFRESH
```

   d. Connect the client certificate to your RACF keyring using the ISPF RACF command:

```
RACDCERT ID(CTGUSER) CONNECT(LABEL('CLIENT.CERT')
RING(CTGSERVERKEYRING) USAGE(CERTAUTH))
```

   The new signer certificate is added to the list in the Signer Certificates view, and can be used by the server to verify the identity of the client application.

You have now configured SSL client authentication.

Configuring the IPIC connection on CICS

To complete this task you use an editor to add a parameter to the startup JCL, you then edit the IPCONN autoinstall user program DFHISCIP, you then use a CEDA command to configure the TCPIPService definition and the IPCONN template definition.

1. Define the system initialization parameter for the key ring by adding the following system initialization parameter to the startup JCL:

   ```
   KEYRING=CTGSERVERKEYRING
   ```

2. Configure an IPCONN autoinstall user program DFHISCIP:
   a. Modify the sample IPCONN autoinstall program to enable the autoinstall of multiple secure IPCONNs.

      CICS provides the IPCONN autoinstall sample program DFHIS*x*IP in Assembler, C, COBOL, and PL/I , where *x* is the program language (A, D, C or P). The sample program does not use a template by default, so if you want autoinstall requests to use a template you must update the program. In this example, the COBOL user program DFHISCIP is updated.

   b. Add the following lines to DFHISCIP to ensure that, when a request arrives from a Java Client with an APPLID beginning with *SSL*, the correct IPCONN template is used to install an IPCONN with the required SSL settings. If the APPLID starts *SSLxxxxx* use the SSLIDP template.

      ```
      IF ISAIC-APPLID(1:3) = 'SSL'
             MOVE 'SSLIDP  ' TO ISAIC-TEMPLATE
             MOVE ISAIC-APPLID TO ISAIC-IPCONN
             PERFORM X000-FINIS.
      ```

   c. Compile and link-edit your program into a data set that can be picked up by your CICS server.

3. Configure a TCP/IP service:
   a. Create the following TCPIPService definition:

      ```
      CEDA  View TCpipservice( SSL51190 )
       TCpipservice   : SSL51190
       GROup          : SSLGROUP
       DEScription    : IPIC LISTENER
       Urm            : DFHISCIP
       POrtnumber     : 51190              1-65535
       STatus         : Open               Open | Closed
       PROtocol       : IPic               IIop | Http | Eci | User | IPic
       TRansaction    : CISS
       Backlog        : 00001              0-32767
       TSqprefix      :
       Host           : ANY
       (Mixed Case)   :
       Ipaddress      : ANY
       SOcketclose    : No                 No | 0-240000 (HHMMSS)
       Maxdatalen     :                    3-524288
       SECURITY
       SSl            : Clientauth         Yes | No | Clientauth
       CErtificate    :
       (Mixed Case)
       PRIvacy        : Supported  |Notsupported | Required | Supported
       CIphers        : 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 WebSphere Application Server" on page 234.

To verify the connection:

1. Enter the following command to run the sample program EciB3. Qualify the location of the SSL key ring, for example ctgclientkeyring.jks, if required:

   ```
   java -DCTG_APPLID=SSLAH com.ibm.ctg.samples.eci.EciB3 local:
   2006 ctgclientkeyring.jks MyPassword
   ```

   The following information is displayed on the screen:

   ```
   CICS Transaction Gateway Basic ECI Sample 3

   Usage: java com.ibm.ctg.samples.eci.EciB3 [Gateway URL]
                                             [Gateway Port Number]
                                             [SSL Keyring
                                              SSL Password]
   To enable client tracing, run the sample with the following Java option:
            -Dgateway.T.trace=on

   The address of the Gateway daemon has been set to local: port 2006
   ```

```
IPIC servers are not listed when running in local mode.
Enter URL of a CICS server, or Q to quit:
```

2. At the prompt, type the following URL: `ssl://`*`lpar`*`:51190`. Where *lpar* is the z/OS LPAR where CICS is running.
3. At the prompt, type a text string to send to the CICS program, for example `my test data`.
4. Type your CICS user ID: `CTGUSER`
5. Type your CICS password.

   The sample program returns verification information, for example:

   ```
   Program EC03 returned 5 containers in channel "SAMPLECHANNEL":
          [CHAR] CICSDATETIME     = 19/05/2010 16:29:31
           [BIT] INPUTDATALENGTH  = 0000000c
          [CHAR] OUTPUTMESSAGE    = Input data was: my test data
          [CHAR] INPUTDATACCSID   = 5348
          [CHAR] INPUTDATA        = my test data
   ```

If the sample program returns `CICS server not found`, this indicates that the SSL connection has not been established. Check the CICS Transaction Server system log for more information, and ensure that the JKS keyring file name and password are correct (the CICS password you entered is not checked because the IPIC connection is configured with Userauth=Identify).

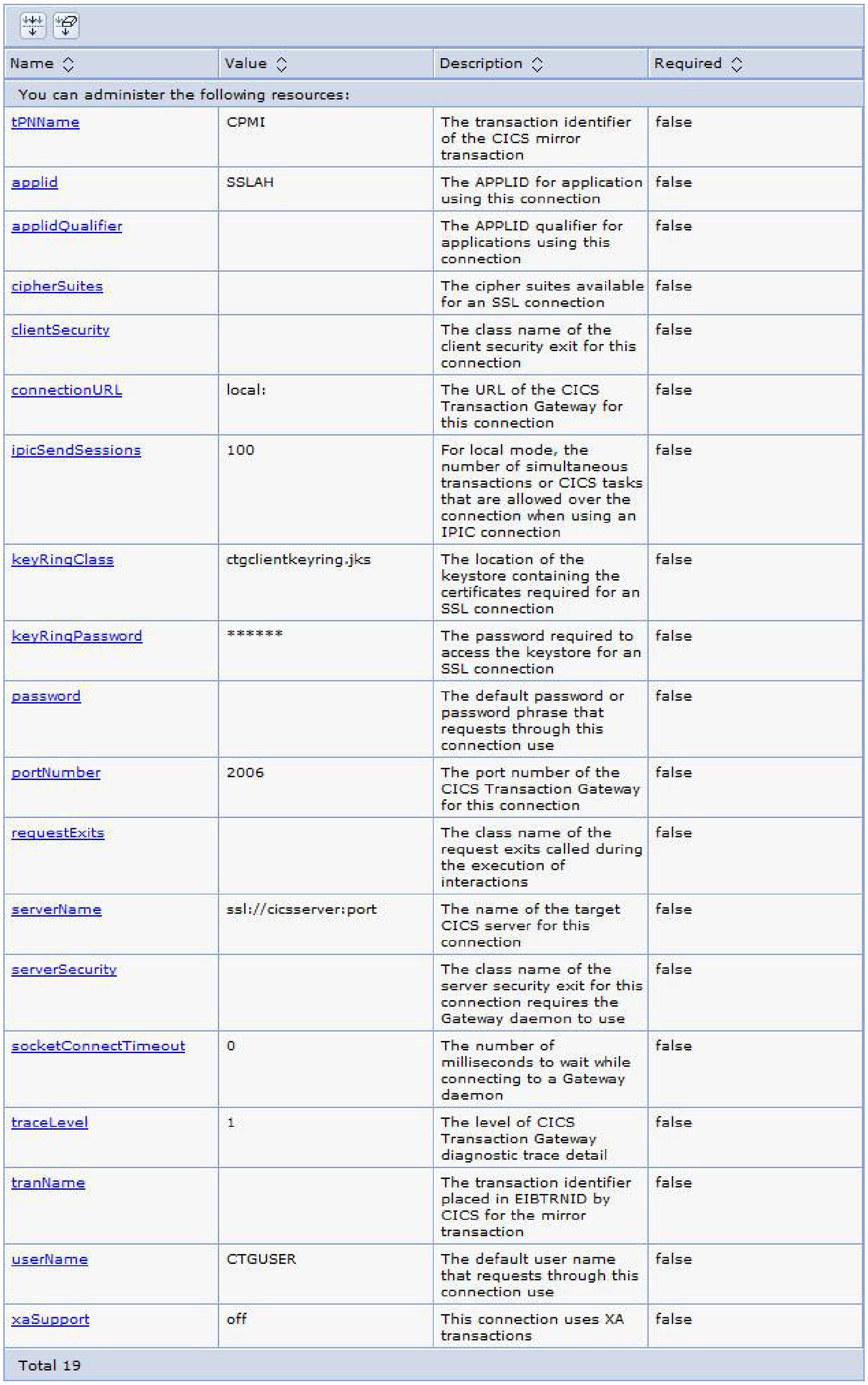
You have now verified the connection.

Configuring WebSphere Application Server

To complete this task you use the WebSphere Application Server Integrated Solutions Console to install the ECI resource adapter, create a connection factory, specify the connection factory properties, and deploy the ECIIVT installation verification test .ear file.

1. Install the CICS Transaction Gateway ECI resource adapter archive (RAR):
   a. In the WebSphere Administrative Console, click **Resources** > **Resource Adapters**, click **Install RAR** .
   b. 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 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 screen:



You can administer the following resources:

Name	Value	Description	Required
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
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	ctgclientkeyring.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

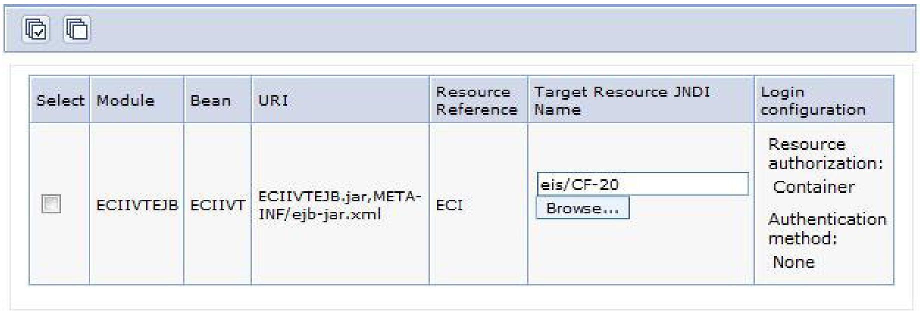
Total 19

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	Bean	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: Container Authentication method: None

c. Save your configuration.
d. Restart WebSphere Application Server if necessary (this depends on the version of WebSphere Application Server you are using).

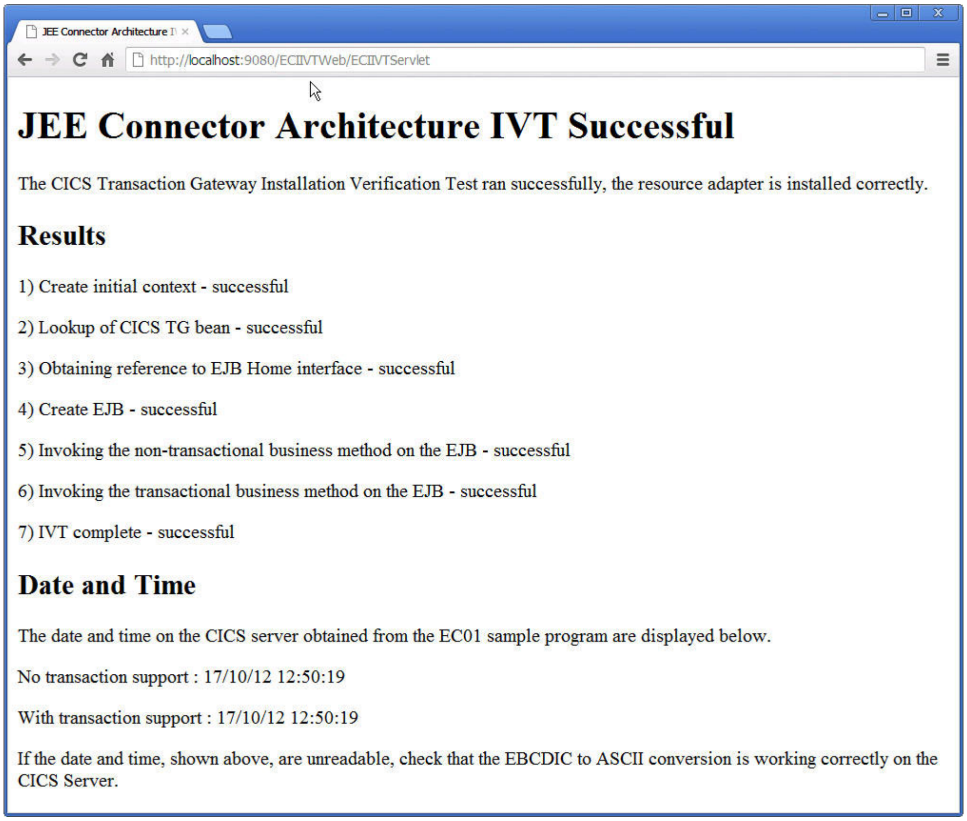
You have now configured WebSphere Application Server.

Testing the SSL scenario

To complete this task you use a browser to go to the ECIIVT web page where you start the ECI resource adapter installation verification test.

1. Open a web browser and enter the following URL:
   `http://localhost:9080/ECIIVTWeb/index.jsp`
2. Click **Run IVT**.

   The JEE Connector Architecture IVT Successful web page is displayed:


JEE Connector Architecture I
http://localhost:9080/ECIIVTWeb/ECIIVTServlet

JEE Connector Architecture IVT Successful

The CICS Transaction Gateway Installation Verification Test ran successfully, the resource adapter is installed correctly.

Results

1) Create initial context - successful

2) Lookup of CICS TG bean - successful

3) Obtaining reference to EJB Home interface - successful

4) Create EJB - successful

5) Invoking the non-transactional business method on the EJB - successful

6) Invoking the transactional business method on the EJB - successful

7) IVT complete - successful

Date and Time

The date and time on the CICS server obtained from the EC01 sample program are displayed below.

No transaction support : 17/10/12 12:50:19

With transaction support : 17/10/12 12:50:19

If the date and time, shown above, are unreadable, check that the EBCDIC to ASCII conversion is working correctly on the CICS Server.

If errors have occurred, run a stack trace by clicking **Stack trace** on the IVT web page. You can also activate CICS Transaction Gateway trace in WebSphere Application Server:

a. From the WebSphere Administrative Console click **Servers** > **Application servers**.

b. Click **server1**.

c. Click **Java and Process Management** > **Process Definition** > **Java Virtual Machine**.

d. In the Generic JVM arguments pane add the following entry:

```
-Dgateway.T=on
```

e. Restart WebSphere Application Server if necessary.

f. Look for the CICS Transaction Gateway trace in the `systemerr.log` file.

You have now completed the scenario.

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.

This scenario uses CICS TG connecting to CICS TS V3.2 over IPIC in remote mode. It uses the default name `ctg.ini` for the configuration file.

Table 20. 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**

Prerequisites

You must satisfy these system requirements.

Here are the system requirements for CICS TS for z/OS:

- The server must be CICS V3.2 or later because IPIC is not available in earlier releases of CICS.
- TCP/IP services must be active in the CICS server.
  - To activate these services, set the 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.
- The TCP/IP network must extend between LPARs if CICS TG for z/OS and the CICS server exist on different LPARs.
- You must set the SEC system initialization parameter to YES to enable security.
- You must have valid RACF user IDs and passwords.

Here are the system requirements for CICS TG:

- CICS TG must be installed.

To test that the scenario works successfully you can use either the supplied samples, or your own applications. If you use the supplied samples, this scenario requires:

- The sample CICS TG server program EC01 must be compiled, defined, and installed on CICS.
- The CICS TG supplied Java sample EciB2 available on the client machine.

Testing your TCP/IP network

At the transport layer, issue ping requests between the operating system that is hosting your CICS TG and the LPAR where your CICS server resides. The ping request response, as shown in the example below, confirms that the TCP/IP communications are working. The ping request also works if CICS TG and the CICS server are not in the same LPAR or if you are using multiple IP stacks on the same LPAR.

```
ping cicssrv2.company.com

Pinging cicssrv2.company.com [1.23.456.789] with 32 bytes of data:

Reply from 1.23.456.789: bytes=32 time<1ms TTL=61
Reply from 1.23.456.789: bytes=32 time<1ms TTL=61
Reply from 1.23.456.789: bytes=32 time<1ms TTL=61
Reply from 1.23.456.789: bytes=32 time<1ms TTL=61

Ping statistics for 1.23.456.789:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Configuring the IPIC server on CICS TG

You must define a server definition for the Gateway daemon to communicate to CICS over IPIC in remote mode.

To define a server definition for the Gateway daemon:

1. Edit the ctg.ini file and define an IPICSERVER definition for your CICS server:
   a. Set HOSTNAME to the name of the z/OS machine that hosts your CICS server.
   b. Set PORT to the port number that your CICS server uses to listen for incoming IPIC requests.

   For example:

   ```
   SECTION IPICSERVER = CICSA
       HOSTNAME=cicssrv2.company.com
       PORT=50889
   ENDSECTION
   ```
2. Save your updated ctg.ini file.
3. Start CICS TG to apply the new IPICSERVER definition.

Configuring the 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      ==> 00001            0-32767
 TSqprefix    ==>
 Ipaddress    ==>
 SOcketclose  ==> No                No | 0-240000 (HHMMSS)
 Maxdatalen   ==>                   3-524288
```

2. Install the CEDA definition.
3. Check that the TCPIPSERVICE is active. On CICS TS, issue the command:

   ```
   CEMT INQ TCPIPSERVICE
   ```

   Check these values:
   - The port number shown is correct.
   - The status shows "Ope" for open.
   - The protocol shown is Ipic.
   - The URM shows the IPCONN autoinstall program, DFHISAIP.

     For example:

     ```
     CEMT INQ TCPIPSERVICE
     STATUS: RESULTS - OVERTYPE TO MODIFY
     Tcpips(SRV50889) Ope Por(50889) Ipic Nos Tra(CISS)
     Con(00000) Bac( 00001 ) 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 11. Operating

When operating CICS Transaction Gateway, it is important to start the product, and the services it uses, in the correct sequence. It is also important to shut down everything in the correct sequence, so that inflight transactions can complete.

Startup and shutdown

The preferred order in which to start CICS, CICS Transaction Gateway and certain services, and optional considerations such as Resource Recovery Services (RRS).

Ways of starting CICS Transaction Gateway

You can start the Gateway daemon in either of the following ways:

- In batch mode using CTGBATCH. For more information about CTGBATCH see "Starting in batch mode" on page 249.
- Using the USS command line interface; see "Starting from a command line" on page 254.

Using CTGBATCH

If you are running in batch mode using CTGBATCH, you create these files:

- ctg.ini (text file)
- ctg.env (STDENV file)

Using the USS command line

If you are using the USS command line method, for example in a test environment, you create these files:

- ctg.ini (text file)
- ctgenvvar (text file)

You can create the files by editing copies of the sample configuration files.

Start command options are available to override some ctg.ini configuration file parameters. See "Options on the ctgstart command" on page 254 for further details.

Startup sequence

Start items in the following order:

1. RRS

   If transactional support is required, RRS must be running on the same z/OS image as the one that CICS Transaction Gateway will use. CICS Transaction Gateway does not support a restart of RRS.
2. TCP/IP

   TCP/IP needs to be running before CICS Transaction Gateway is started. CICS Transaction Gateway supports a restart of TCP/IP, although no requests can be serviced until the stack is restarted.
3. CICS

Consider running CICS and CICS Transaction Gateway in the same z/OS image, to avoid EXCI pipes using slots in the XCF (cross-system coupling facility) group in the sysplex couple data set.

If transactional support is required, and the EXCI protocol is used to communicate with CICS, CICS and CICS Transaction Gateway must be in the same z/OS image.

CICS Transaction Gateway supports a restart of CICS, but requests fail with an ECI_ERR_NO_CICS error until CICS is available again.

4. Gateway daemon

   If an automation tool requires the console to listen and write from the same address space as the invoking executable, set environment variable _BPX_SHAREAS to YES. This causes the Gateway daemon to run in the same address space as CTGBATCH.

Shutdown sequence

The system and its components must be shut down in this order:

1. Shut down the Gateway daemon normally, allowing all transactions that are in-flight to complete.
2. Shut down TCP/IP normally.
3. Shut down CICS normally.
4. Terminate CTGRRMS.
5. Shut down CTGRRM normally.

Starting CICS Transaction Gateway

CICS Transaction Gateway can be started either as a batch job (the recommended option) or from the command line.

- Submit JCL to start CICS Transaction Gateway as a batch job if you want to register with the ARM (Automatic Restart Manager) component of MVS. For more information see “Automatic restart management” on page 246. ARM is the recommended option.
- Alternatively you can start CICS Transaction Gateway from the z/OS UNIX System Services command line.

When you start the CICS Transaction Gateway, it reads configuration information from ctg.ini, ctgenvvar, and stdenv. When you issue the ctgstart command, you can specify startup options to override this configuration information. These options and their values are described in “Options on the ctgstart command” on page 254.

If XA support is enabled, the recoverable resource management services (RRMS) of the z/OS sync point manager are required.CICS Transaction Gateway accesses RRMS through CTGRRMS services which run in a separate address space. This address space is started in these circumstances:

- The first time you start the Gateway daemon.
- When you run the `ctgasi` command; see “Starting, stopping or refreshing the CTGRRMS services” on page 128.

The CTGRRMS address space remains for the life of the z/OS image. See “Enabling CTGRRMS services” on page 127 for information about CTGRRMS services. See “Starting, stopping or refreshing the CTGRRMS services” on page 128 for information about how to start, stop and restart CTGRRMS services.

A Gateway daemon cold start option startup is provided. Cold start is for use with XA transactions and ensures residual records in RRS are cleaned up after failure of a transactional component, or by manual intervention by an operator through the RRS panels. For more information see “Cold start.”

Cold start

A Gateway daemon cold start option is provided to resolve heuristically-completed transactions. A heuristic error might occur after the failure of a transactional component, or manual intervention by an operator through the RRS panels.

In normal operation, the XA transaction manager component is responsible for issuing requests to “forget” transaction branches that have been heuristically resolved. If there is a heuristic error and the transaction manager is still running, it issues a “forget” call directly. However, if there is a heuristic error and the transaction manager is restarted before it has time to issue a “forget” call, the following happens:

- The transaction manager issues a “recover” call on startup.
- The recover returns a list of prepared and heuristically-completed transactions.
- The transaction manager issues “forget” calls as appropriate.

In exceptional cases, the XA transaction manager might not have a record of a transaction branch being in “InForget” state, whereas RRS still has a record of the transaction branch being “InForget”, “InCommit” or “InBackout” state. In this situation, the XA transaction manager does not issue an XA “forget” call, even if the Unit of Recovery (UR) is returned in response to an XA recover flow. This can happen:

- If the XA transaction manager is unable to issue the XA “forget” call and is timed out after multiple attempts (if the transaction manager issues a “forget” call to the Gateway, the “forget” call fails, the transaction manager repeats this and eventually times out issuing the “forget” call).
- If RRS recreates committed or backed out units of recovery (UR) during CICS Transaction Gateway restart processing (when RRS is restarted, or if a UR has been completed by a different Gateway daemon with the HA group).

The Gateway daemon indicates the number of recovered URs during initialization by issuing message CTG8628I. If this number is greater than zero, review the outstanding URs through RRS and consider restarting the Gateway daemon using the cold start option to remove those URs in “InForget” state. If a cold start is not specified (normal start), CICS Transaction Gateway does not issue any forget calls and the URs remain visible in RRS.

URs in “InCommit” or in “InBackout” state have been resolved by a Gateway daemon in the HA group on behalf of another Gateway daemon, following a failure during two-phase commit processing. Such URs progress to “InForget” state, when the resolving Gateway daemon is shut down. These URs are cleared when the originating Gateway daemon performs a cold start. The statistic GD_LXACOMP indicates when a Gateway daemon has resolved XA transactions on behalf of another Gateway daemon in the HA group.

Multiple address spaces

When running the CICS Transaction Gateway, multiple address spaces can be used for running the individual processes involved in starting the Gateway daemon. If you want these processes to run in a single address space, take the following actions:

- Set the _BPX_SHAREAS environment variable in STDENV to either YES or MUST.
- Also ensure that the extended attribute for the shared address space on the ctgstart script has not been unset.
  1. To display the extended attributes for files, use USS command *ls -E*
  2. To set the shared address space attribute on files use USS command *extattr +s*

The following table shows the results of different combinations:

extattr setting	**_BPX_SHAREAS setting**	**Result**
+s	YES	Same address space
+s	MUST	Same address space
+s	NO	Separate address space
-s	YES	Separate address spaces
-s	MUST	CTGBATCH fails to start the Gateway daemon
-s	NO	Separate address spaces
(any)	Not set	Warning, separate address space

If you use CTGBATCH to run the Gateway daemon, and you have defined the environment variable AUTH_USERID_PASSWORD=YES to enforce user ID and password authentication, set environment variable _BPX_SHAREAS to YES and ensure that the SCTGLOAD library, and all other libraries in the STEPLIB have been defined as program controlled. For more details see "Configuring for client certificate mapping" on page 143.

If you run the Gateway daemon from USS, and you have defined the environment variable AUTH_USERID_PASSWORD=YES, ensure that environment variable _BPX_SHAREAS is set to NO. This forces the Gateway daemon to run in a new program controlled address space.

Related information:

"Environment variables: local and remote mode" on page 100
Environment variables available for use with local mode and remote mode topologies.

Automatic restart management

Use the Automatic Restart Manager (ARM) to restart the Gateway daemon if a failure occurs.

If you start the CICS Transaction Gateway with JCL (see "Starting in batch mode" on page 249), you can register the job with z/OS ARM. This allows z/OS to restart the Gateway automatically if it fails, or if the system on which it was running fails.

Automatic restart management is a sysplex-wide integrated automatic restart mechanism that:

- restarts a subsystem in place if an abend occurs (or if a monitor program notifies ARM of a stall condition)
- restarts CICS data sharing servers in the event of a server failure
- restarts a failed z/OS image

The main benefits of ARM are that it:

- Eliminates the need for operator-initiated restarts, or restarts by other automatic packages, thereby:
  - Improving emergency restart times
  - Reducing errors
  - Reducing complexity
- Provides cross-system restart capability. It ensures that the workload is restarted on z/OS images with spare capacity, by working with the z/OS workload manager.
- Allows all elements within a restart group to be restarted in parallel. Restart levels (using the ARM WAITPRED protocol) ensure the correct starting sequence of dependent or related subsystems.

For more information about Automatic Restart Management, see the relevant version of the publication *z/OS MVS Setting up a Sysplex*, SA22–7625.

See the SCTGSAMP PDS library for sample JCL. The job steps relating to CTGARM are commented-out, so that users who do not use ARM can run the samples.

Prerequisites:

Use the CTGARM utility to register the CICS Transaction Gateway. This utility is supplied in the load library SCTGAUTH. It is used to register and deregister with ARM, around the Gateway daemon JCL step, which invokes the `ctgstart` script through CTGBATCH.

To use ARM you need ARM and a Sysplex controller. You need to define policies for automatic restart management; see *z/OS MVS Programming: Sysplex Services Guide*, SA22-7617. If XA support is enabled, create a policy that restarts the CICS Transaction Gateway in the same z/OS image as the CICS server.

The SCTGAUTH load library must be defined as APF-authorized. To do this dynamically, use a command like the following:

```
SETPROG APF,ADD,dsname=xxxxxxx,volume(xxxxxx)
```

To register the library permanently, include an entry in the appropriate SYS1.PARMLIB member 'PROGxx'.

The CTGARM utility takes one of the following parameters:

- `'R[egister] ARM_ID [ARM_TYPE]'`
- `'D[eregister]'`

CTGARM messages are written to the SYSPRINT DD destination, which must be defined in the calling JCL step. If SYSPRINT is not defined, the step will fail.

ARM_ID is a unique 16–character ID. ARM_TYPE is an 8–character restart level; the default is SYSLVL2. Valid characters for both ARM_ID and ARM_TYPE are:

- Uppercase alphabetic characters
- The numbers 0 through 9
- $, #, @, and underscore (_).

The first character might not be a number.

CTGARM returns the following codes:

0 No errors.

4 Restarting.

8 Miscellaneous error, including already registered or not unique ID.

12 Severe error, for example ARM is not installed.

For ARM error codes see *z/OS MVS Programming: Sysplex Services Reference*, SA22-7618.

System automation messages:

Startup and shutdown messages, which could be used with a systems management program such as IBM Tivoli System Automation, are logged to the z/OS console. Consult the documentation supplied with your systems management program for details of how to use these messages in its configuration.

Messages

Table 21. Messages logged to the z/OS console on CICS Transaction Gateway startup and shutdown

Messages	Event
CTG6400I CICS Transaction Gateway is starting	The CICS TG is initializing, following a startup request.
CTG6405E The Gateway daemon has terminated abnormally	The CICS TG has terminated abnormally.
CTG6420I Health value has been reset to 100	Health value has been reset to 100.
CTG6421W Health value has been set to 0	Health value has been set to 0.
CTG6490I Normal shutdown of Gateway daemon started by *user or systems management program*	The CICS TG is initializing a shutdown.
CTG6509I Immediate shutdown of Gateway daemon started by *user or systems management program*	The CICS TG is initializing a shutdown.
CTG6511I Gateway daemon has shut down	The CICS TG has shut down.
CTG6512I CICS Transaction Gateway initialization complete	The CICS TG startup is complete; ready for work.
CTG6513E CICS Transaction Gateway failed to initialize	The CICS TG initialization process started, but could not be completed. Operator intervention is required.
S806 abend	EXCI load library incorrect. Unable to initialize JNI.

When the messages in this table are output to the z/OS console, the jobname of the Gateway daemon is also shown, after the message identifier:

```
<CTGnnnnI> <JOBNAME> <TEXT>
```

ATR229D CANCEL DELAYED

The following WTOR is written to the system log when XA support is enabled and a network connection is broken when a transaction is in doubt:

```
0000 *<NN> ATR229D CANCEL DELAYED. REPLY WAIT, BACKOUT, OR COMMIT TO
RESOLVE
      INDOUBT UR. URID = <UR identifier>
```

If a network connection is broken between the application server, that is WebSphere, and the Gateway daemon when a unit of recovery is in doubt the message above is written to the system log. When WebSphere reconnects to the Gateway daemon, or any Gateway daemon in the group, and issues a commit or backout request the transaction is resolved, the WTOR is cancelled and the following message is written to the log:

```
10:45:09.23         00000010  IEE400I THESE MESSAGES CANCELLED - <NN>
```

The connection manager thread which was dealing with the network connection that failed stays blocked until the unit of recovery is resolved.

It is recommended that you do not reply to this message and that you allow WebSphere to reconnect to a Gateway daemon in the group and resolve the transaction. When the transaction is resolved the WTOR is cancelled and does not need a response. Do not use an automated systems management system to respond to the ATR229D.

Related information:

"Multiple address spaces" on page 245
When running the CICS Transaction Gateway, multiple address spaces can be used for running the individual processes involved in starting the Gateway daemon. If you want these processes to run in a single address space, take the following actions:

"Automatic restart management" on page 246
Use the Automatic Restart Manager (ARM) to restart the Gateway daemon if a failure occurs.

"Startup and shutdown" on page 243
The preferred order in which to start CICS, CICS Transaction Gateway and certain services, and optional considerations such as Resource Recovery Services (RRS).

Starting in batch mode

The recommended way of running the CICS Transaction Gateway for z/OS Gateway daemon as a production system is in batch mode. CTGBATCH is a utility, used to launch USS programs through the MVS batch environment and route stdout and stderr I/O to MVS destinations.

The CTGBATCH batch mode program provides the ability to write log messages to the following destinations from the CICS Transaction Gateway for the Gateway daemon:

- The JES log
- An HFS file

Define the log message destinations by including STOUT and STDERR DD statements in the CTGBATCH job step. Do not use an MVS sequential data set for log messages.

CTGBATCH is used primarily to start the Gateway daemon by invoking the USS script ctgstart. To successfully run the ctgstart script some configuration environment variables need to be in place. CTGBATCH parses a STDENV file to set environment variables used by the Gateway daemon. The STDENV file is defined by DD statement STDENV on the CTGBATCH job step.

If you are an existing user, use the conversion tool ctgconvenv to convert the ctgenvvar script to the STDENV file. For information about the STDENV file see STDENV file.

A comprehensive sample is supplied in the SCTGSAMP library. Sample JCL for CTGBATCH can be found in the SCTGSAMP library as members CTGJOB and CTGPROC.

CTGBATCH considerations:

Things to consider when using CTBATCH to launch USS programs through the MVS batch environment include the location of message logs and the national language for messages.

Start CTGBATCH using a JCL step in the following format:

```
//CTGBATCH  EXEC PGM=CTGBATCH,
            PARM='<LE runtime options><fully qualified HFS path to target
            executable><parameters>'
```

where:

<LE runtime options>
: Is a free-form string of Language Environment options, terminated by a slash (/) character. At the very least, the slash (/) must be included to indicate an empty set of options.

<fully qualified HFS path to target executable>
: Explicitly represents the target program. For example,

`'/usr/lpp/cicstg/ctg700/bin/ctgstart'`

<parameters>
: Is the parameter string to be supplied to the target program. For example,

`'-noinput -x'`

Note: There must be a space between the target executable and the <parameters> string.

Therefore, a JCL step using CTGBATCH to invoke the ctgstart script is as follows:

```
//CTGBATCH  EXEC PGM=CTGBATCH,
//          PARM='//usr/lpp/cicstg/ctg700/bin/ctgstart -noinput'
```

An important difference between CTGBATCH and BPXBATCH is that the Language Environment options override was not part of the BPXBATCH PARM string syntax. If a BPXBATCH PARM string is used, unchanged, with CTGBATCH, everything preceding and including the first slash (/) character is processed as Language Environment options, and the rest as CTGBATCH parameters. The likely result is message "CTG0828E CTGBATCH The target executable 'target-program' does not exist".

Note:

1. The PARM parameter in the JCL EXEC PGM= statement is limited to a string value of 100 characters (this applies to BPXBATCH aswell). See “Environment variables: remote mode” on page 102 for details on using the CTGSTART_OPTS environment variable to circumvent this limitation with ctgstart parameters.
2. When CTGBATCH is used to start the CICS Transaction Gateway for z/OS daemon via the ctgstart script, the '-noinput' switch must be specified to enable the TSO-SDSF systems management function

CTGBATCH accepts the following DD statements:

- STDOUT

- STDERR
- STDENV
- CTGDBG

and one of the following national language support DD statements:

- CTGMSGEN
- CTGMSGJA
- CTGMSGZH

STDOUT and STDERR DD statements:

To route log messages to a particular destination, the JCL must contain DD cards STDOUT and STDERR in the DD statements which mean route stdout and stderr to the specified DD location.

DD cards STDOUT and STDERR must be writable destinations. Valid options are:

- SYSOUT=x for JES logs
- DSN=<MVS sequential data set>,DISP=SHR
- DSN=<MVS partitioned data set member>,DISP=SHR
- PATH=<HFS file> PATHMODE=<mode> PATHOPTS=<options>

DD cards STDOUT and STDERR are defined as writable HFS files, as shown in the following examples, to be created if they do not already exist (OCREAT), appended if they do exist (OAPPEND), and opened for writing only (OWRONLY). Read-write permissions are given to the batch user ID (SIRUSR,SIWUSR) with read-only to all other users (SIRGRP,SIROTH).

```
//STDOUT  DD PATH='/u/ctgusr/ctglogs/ctg1_stdout.log',
//   PATHOPTS=(OCREAT,OAPPEND,OWRONLY),
//   PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIROTH)
//STDERR  DD PATH='/u/ctgusr/ctglogs/ctg1_stdoerrlog',
//   PATHOPTS=(OCREAT,OAPPEND,OWRONLY),
//   PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIROTH)
```

If the STDOUT DD statement is omitted, a JES log with the name 'SYS00001' will be dynamically allocated the first time that data is written to the stdout destination.

If the STDERR DD statement is omitted, a JES log will be dynamically allocated. The name is dependent upon the MSGFILE Language Environment runtime option. If allowed to default the name will be 'SYSOUT'.

Note: The location specified in the STDERR DD statement will also contain remote mode JNI log messages.

STDENV DD statement:

DD card STDENV must be a readable source. If no STDENV DD statement is defined in the JCL, CTGBATCH will still attempt to run the target program specified in the PARM string. This is acceptable if the target program is itself a self-contained executable program, for example, ctgasi.

Valid options are:

- SYSIN to have the STDENV data inline with the JCL
- DSN=<MVS sequential data set>,DISP=SHR
- DSN=<MVS partitioned data set member>,DISP=SHR

- PATH=<HFS file> PATHMODE=<mode> PATHOPTS=<options>

For example, the following DD card STDENV is defined to be opened for reading only (ORDONLY) with read-only permissions for the batch user ID (SIRUSR).

```
//STDENV  DD PATH='/u/ctgusr/ctgcfg/my_ctg1.env',
//   PATHOPTS=(ORDONLY),PATHMODE=SIRUSR
```

CTGDBG DD statement:

If you include a dummy DD statement CTGDBG in the CTGBATCH job, extra runtime environment and status log messages are generated.

These messages are written to the STDOUT and STDERR destinations as appropriate. This facility can be used as an aid to resolving upgrade problems and also for problem determination.

National language support DD statements:

Inclusion of one dummy DD statement from CTGMSGEN (English), CTGMSGJA (Japanese) or CTGMSGZH (Chinese) in the CTGBATCH job will determine which of the three national language message bundles is used by CTGBATCH. This does not affect the national language resources of the Gateway daemon itself.

If no national language support DD statements are defined in the CTGBATCH JCL, the default of CTGMSGEN (English) is assumed. If more than one of the valid three national language support DD statements are defined, CTGBATCH fails with message CTG0832E and JES return code 20.

z/OS considerations:

To make CTGBATCH and the Gateway daemon run in the same address space, set the _BPX_SHAREAS environment variable in STDENV to either YES or MUST.

The recommended value for the Gateway daemon is YES. If you set it to NO, they run in separate address spaces. If you do not set it, CTGBATCH issues a warning, and they run in separate address spaces. See “Multiple address spaces” on page 245 for information about problem determination with multiple address spaces.

To run CTGBATCH and the Gateway daemon in a single address space with security active, CTGBATCH needs to be program controlled. For information about program control see “Security error due to surrogate checking problem” on page 283.

The setting of _BPX_SHAREAS can affect the JOBNAME that the user must specify when issuing system management commands to the Gateway daemon. If the JOBNAME is less than 8 characters in length, and the Gateway daemon runs in a separate address space, the JOBNAME to use for system management commands via TSO/SDSF is the base JOBNAME with a numeric suffix.

Region size considerations:

Set the REGION parameter on the EXEC card in the JCL according to the virtual storage requirements for thread and Java heap settings in the Gateway daemon.

For more information about calculating the required setting see Avoiding out of memory conditions.

To verify the actual region size allocated for the CICS Transaction Gateway, include the CTGDBG DUMMY DD statement in the JCL step for CTGBATCH. Message CTG0813I shows the current region size.

The following parameters can also limit the values that you can choose for the REGION parameter:

1. The ASSIZEMAX parameter of the OMVS segment of a RACF user ID; for more information see the *RACF Command Language Reference*.
2. The UNIX System Services MAXASSIZE parameter found in SYS1.PARMLIB(BPXPRMxx); for more information see the *UNIX System Services Planning*. The value specified for ASSIZEMAX overrides any value provided by the MAXASSIZE parameter

Note: Use of REGION=0M. The actual region size available when REGION=0M is used is unpredictable, due to optional implementation of the system exit routine IEFUSI. REGION=0M indicates that the address space must be given as much memory as possible. However, large REGION sizes might be policed by the IEFUSI exit, and reduced to a default size. If this default size is too small the CICS Transaction Gateway is likely to fail with out of memory problems.

CTGBATCH examples:

Example JCL follows that enables log messages to be written to the various destinations and various sources for the STDENV data.

Writing messages to the JES logs:

This JCL snippet shows how to write messages to the JES logs STDOUT and STDERR and read the environment variables from the HFS file '/u/ctgusr/stdenv.txt'.

```
//STDOUT  DD SYSOUT=*
//STDERR  DD SYSOUT=*
//STDENV  DD PATH='/u/ctgusr/stdenv.txt',
// PATHOPTS=(ORDONLY),PATHMODE=SIRUSR
/*
```

Writing messages to HFS files:

This example JCL snippet shows how to write all stdout and stderr data to HFS files and store the environment variables in-line.

```
//STDENV  DD *
PATH=/bin:/usr/lpp/java/bin
_BPX_SHAREAS=YES
AUTH_USERID_PASSWORD=YES
CICSCLI=/u/ctgusr/cfg/myctg1.ini
/*
//STDOUT  DD PATH='/u/ctgusr/ctglogs/ctg1_stdout.log',
//   PATHOPTS=(OCREAT,OAPPEND,OWRONLY),
//   PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIROTH)
//STDERR  DD PATH='/u/ctgusr/ctglogs/ctg1_stdoerrlog',
//   PATHOPTS=(OCREAT,OAPPEND,OWRONLY),
//   PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIROTH)
//*
```

JES return codes:

The following list contains JES return codes and their meanings.

0 OK The target program was started successfully.

4 WARNING_STATE
The target program was started successfully, but CTGBATCH found some potential problems. See log messages for details.

8 SPAWN_FAILURE
The target program did not start.

12 LOGINIT_FAILURE
CTGBATCH failed to open the defined DD statement and the dynamically allocated JES destination was also unavailable.

16 PIPEINIT_FAILURE
CTGBATCH failed to create the interprocess pipes required for communicating with the target program.

20 INITSTATE_FAILURE
CTGBATCH initialization failed. See log messages for details.

24 LOG_WRITE_FAILURE
CTGBATCH log write failure. During run time, CTGBATCH did not write to one of the defined STDOUT or STDERR log destinations. A subsequent attempt to write data to the default Language Environment stdout or stderr destination also failed.

Starting multiple CICS Transaction Gateways

See the SCTGSAMP PDS library for sample JCL.

To start multiple CICS Transaction Gateways from OMVS with console input when directed to a port that is portshared, define the jobname "OMVS" as having access to the shared port in the TCP/IP profile. If console input is not required, then start the CICS Transaction Gateways via JCL and define the JCL jobname in the TCPIP profile.

See “Port is in use by another task” on page 274, for information on a problem that can be caused when the CICS Transaction Gateway jobname is shorter than eight characters.

Starting from a command line

To start the CICS Transaction Gateway with the default options, type `ctgstart` at the command line and press Enter.

A Gateway console session starts, and messages are displayed showing the values being used. To override the startup defaults, type: `ctgstart` at the command line, followed by the startup options you require, and press Enter. A Gateway console session starts, and messages are displayed showing the values being used.

To get help for the startup options, enter: `ctgstart -?`

Options on the ctgstart command:

To override the startup defaults, from the command line enter ctgstart at the command line, followed by the required startup options.

Option	Purpose
-applid	The Gateway daemon applid.
-applidqualifier	The Gateway daemon applid qualifier.

Option	Purpose
-classpath=classpath	Additional classpath entries to use when launching the JVM. For example, the location of a jar file containing request exits.
-dnsnames	Displays symbolic TCP/IP host names in messages.
-dumpoffset=*offset*	The offset from which displays of any data blocks start. If the specified offset is greater than the total length of data, the data is dumped as if the offset were 0.
-initconnect=*number*	The initial number of connection manager threads.
-initworker=*number*	The initial number of worker threads.
-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 switch.
-keyring=*keyring*	The SSL key ring path and file name.
-keyringpw=*keyringpw*	The SSL key ring password. For example: `ctgstart -sslport=port_number -keyring=keyring -keyringpw=keyringpw` An error message is generated if the *keyringpw* parameter is used on its own without the corresponding *keyring* parameter in the ctgstart - command line.
-maxconnect=*number*	The maximum number of connection manager threads. If this value is set to -1, no limits are applied to the number of connection manager threads.
-maxworker=*number*	The maximum number of worker threads. If this value is set to -1, no limit is applied to the number of worker threads.
-noinput	Disables the reading of input from the console. If this option is selected you cannot stop the Gateway by input to the console session. See "Stopping the CICS Transaction Gateway" on page 256.
-port=*port_number*	The TCP/IP port number assigned to the CICS Transaction Gateway
-stack	Enables exception stack tracing but no other type of tracing. All checked Java exceptions are traced, including exceptions that are expected during typical operation of the CICS Transaction Gateway.
-start=*cold*	Ensures that CICS Transaction Gateway issues a call to RRS to forget each unit of recovery associated with the Gateway daemon which is in 'in-forget' state in RRS. If you specify this value, the forget calls are completed before any requests are processed by the Gateway daemon. If this option is not specified, the default CICS Transaction Gateway startup logic is used and the forget calls are not issued to RRS. For more information see "Cold start" on page 245.
-statsport=*port_number*	The TCP/IP port on which the Gateway daemon listens for statistics API requests.
-sslport=*port_number*	The TCP/IP port on which the Gateway daemon listens for SSL requests.

Option	Purpose
-trace	The standard trace option . If this option is selected the first 80 bytes of the `COMMAREA` are traced by default. For more information see Tracing. Trace output is written to stderr.
-truncationsize=*nnn*	The maximum size of any data blocks shown in the trace, where *nnn* is any positive integer. You can use this option in addition to either -trace or -x to override the default size set by these options. For example, to switch on standard tracing and dump a maximum of 20,000 bytes: `ctgstart -trace -truncationsize=20000` If you specify a value of 0, no data blocks are shown in the trace.
-tfile=*pathname*	Writes trace messages to the file specified in *pathname* if tracing is enabled. This option overrides the default destination for trace output (see the **-trace** option).
-tfilesize=*number*	The maximum size of the trace text file in KB.
-x	Enables full debug tracing. Full debug tracing includes everything traced by the -trace option, with additional information including information about the internal workings of the CICS Transaction Gateway. This option traces the entire COMMAREA by default. This option significantly decreases performance.

Stopping the CICS Transaction Gateway

There are two scenarios for shutdown, normal and immediate.

The normal shutdown scenario is the recommended method for stopping the CICS Transaction Gateway. The immediate shutdown scenario is not recommended except in emergencies; there is no guarantee that work in progress will complete.

Normal shutdown

In a normal shutdown, the Gateway daemon waits for work in progress to complete. During this time, new work is not allowed to start. When all work has completed or all Java clients have disconnected, the Gateway daemon shuts down.

Immediate shutdown

Any outstanding work is terminated abruptly. Existing connections are broken; requests for new connections are refused. The Gateway daemon shuts down without waiting for XA transactions to complete. Extended LUWs and work in a pre-prepare state are rolled back.

XA transactions that are in post-prepare state are not rolled back during an immediate shutdown. If no other instance of the CICS Transaction Gateway is running under the same RRM name, it is not possible for the transaction manager (WebSphere) to recover indoubt XA transactions until the CICS Transaction Gateway is restarted.

If TCP/IP load balancing is in use, multiple CICS Transaction Gateways are running under the same RRS name. Requests to resolve indoubt XA transactions are sent to other Gateway daemons that are still running in the same Gateway group.

You can perform an immediate shutdown if a normal shutdown is taking too long. You cannot request a normal shutdown after you have issued the command for an immediate shutdown.

Normal shutdown

When CICS Transaction Gateway shuts down normally, transactions that are already in progress are completed.

- All one-phase-commit transactions and XA transactions are allowed to complete as normal.
- No new transaction requests are accepted.
- No new XA recover requests are accepted.

Stop the Gateway daemon, as described in "Stopping a Gateway daemon."

Immediate shutdown

Immediate shutdown of CICS Transaction Gateway forces the abnormal termination of transactions that are already in progress.

This method of shutdown is not recommended, for the following reasons:

- There is no guarantee that any work will complete.
- There is no guarantee that any tidy-up is performed.

During an immediate shutdown, the following guidelines apply:

- Transactions already in progress are terminated abnormally.
- All in progress one-phase-commit LUWs are backed out (rolled back) and not recovered.
- For XA transactions:
  - All pre-prepare work is rolled back.
  - All post-prepare work is left in RRS to be recovered either upon restart, or by another CICS Transaction Gateway in the same Gateway group.

Stop the Gateway daemon, as described in "Stopping a Gateway daemon."

Stopping a Gateway daemon

If you started the CICS Transaction Gateway from the command line, and you did not specify the `-noinput` parameter, you can stop the Gateway by typing the correct character and pressing the `Enter` key in the Gateway console session.

Accepted characters are as follows:

Usual shutdown
`Q` or `-`

Immediate shutdown
`I`

If you did not start the CICS Transaction Gateway with the **-noinput** option, messages describing which keys to use for shutdown are displayed at the end of startup. If you used the `-noinput` option, you must kill the process to stop the CICS Transaction Gateway.

If you used JCL to start the CICS Transaction Gateway, use one of the supported MVS system commands to shut down the Gateway. See Gateway daemon administration for details.

Note: When you cancel the main CICS Transaction Gateway job, all other jobs are cancelled. For example, for a jobname CICSTGQ, the jobs named CICSTGQ1 through CICSTGQ*n* would also be cancelled.

If you have problems stopping the Gateway daemon, see fails to shut down.

Gateway daemon administration

Gateway daemon administration tasks include starting and stopping, tracing, displaying statistics and obtaining JVM dumps.

If each Gateway daemon runs as a separate MVS address space, with a unique job name, you can complete the following tasks using MVS system commands:

- Set the Gateway daemon trace
- Set JNI trace
- Query trace settings
- Stop the Gateway daemon
- Display statistical information about the CICS Transaction Gateway.
- Obtain a Java heap dump, a system dump, or a Java dump from a running JVM.
- Obtain dumps containing information about the CICS Transaction Gateway configuration, and the current JVM.

Shutting down

To perform a typical shutdown, issue one of the following MVS system commands.

- `/P  JOB_NAME`
- `/F JOB_NAME,APPL=SHUTDOWN|SHUT`

where `JOB_NAME` is the name of the CICS Transaction Gateway job that you want to send the command to.

You can use either SHUTDOWN or SHUT command. If the command is misspelled, the command is rejected, and a message issued.

If the command is not acceptable because an immediate shutdown has already been issued, the request is ignored.

Shutting down immediately

If the command is successful, a message is displayed, and written to the system log.

To perform an immediate shutdown, issue the following MVS system command:

```
/F  JOB_NAME,APPL=SHUTDOWN|SHUT,IMMEDIATE|IMM
```

where `JOB_NAME` is the name of the CICS Transaction Gateway job that you want to send the command to. You can use either of the following pairs of commands:

SHUTDOWN or SHUT
IMMEDIATE or IMM

If the command is misspelled, the command is rejected, and a message is issued.

Controlling trace

Trace is controlled through commands issued from the administration interface.

Commands are provided for the following tasks:

- Set the trace level for the Gateway daemon
- Set the trace file size limit
- Set the amount of data from the hex dump to trace
- Set the offset within a hex dump
- Do a full hex dump
- Turn JNI trace on or off
- Query trace settings

Use the MVS MODIFY command `/F` *JOB_NAME*,APPL=TRACE, together with appropriate options, to control trace. Some examples of controlling trace follow. See Trace options for an explanation of the options.

Querying trace settings

Query trace settings to see values such as the trace file size limit or the amount of data to trace.

To query trace settings, enter the following command:

```
/F JOB_NAME,APPL=TRACE
```

Setting the Gateway trace

Use this command to set the trace level to full debug.

To set the level to full debug tracing, enter the following command:

```
/F JOB_NAME,APPL=TRACE,TLEVEL=4
```

To set a DUMPOFFSET of 128 and TRUNCATIONSIZE of 512, enter the following command:

```
/F JOB_NAME,APPL=TRACE,DUMPOFFSET=128,TRUNCATIONSIZE=512
```

Setting the JNI trace

Use these commands to set the level of the Java Native Interface (JNI) trace, and to disable the trace.

To enable JNI trace, enter the following command:

```
/F JOB_NAME,APPL=TRACE,JNILEVEL=1
```

To disable JNI trace, enter the following command:

```
/F JOB_NAME,APPL=TRACE,JNILEVEL=0
```

Combining options

Use multiple options on a single command to specify the required tracing characteristics.

For example, the following command sets the trace level for the Gateway daemon to 2, specifies a full data dump, and enables JNI trace:

```
/F JOB_NAME,APPL=TRACE,TLEVEL=2,JNILEVEL=1,FULLDATADUMP
```

Controlling health monitoring

You can find the current health status of your system, and reset the health status to 100.

Determining health status

The current health status is available in the GD_CHEALTH statistic. This information describes how to find the current health status.

Procedure

1. Log on to TSO and navigate to the SDSF console.
2. Enter the following MVS system command: `/F <JOB>,APPL=STATS,GS=GD_CHEALTH` where `<JOB>` is the JCL jobname of the Gateway daemon. A message tells you the current value for health status.

Related concepts:

"Health reporting" on page 155
The Gateway daemon can monitor certain error codes to determine the health of communications with CICS.

Related information:

"Resetting health status"
This information describes how to reset the health status to 100.

Resetting health status

This information describes how to reset the health status to 100.

Communication problems can cause the health status to drop. If the status drops to 0, it might have to be reset before the load balancer will send any new connections to the affected Gateway daemon. The health status can be reset regardless of the current health status value; you do not have to wait until the status drops to 0.

The health status can recover without intervention in some circumstances. For example, if a Java client application has 21 established connections to the Gateway daemon, and an ECI request is sent across each of the first 20 connections and fails with ECI_ERR_NO_CICS, health drops to 0. No new connections can be established, but because the Java client application still holds an existing connection to the Gateway, a successful request sent over that connection will cause the health to recover.

1. Log on to TSO and navigate to the SDSF console.
2. Enter the following command: `/F <JOB>,APPL=HEALTH,RESET` where `<JOB>` is the JCL jobname of the Gateway daemon. A message tells you whether the command completed successfully.

Administration options

Options are available for shutdown, tracing, statistics, and dumps. An option is also available for dynamic interaction with request monitoring exits.

Shutdown options

This option is available for use with the MVS system command `/F <JOB_NAME>,APPL=SHUT`.

To get help on these options, issue the following command:

```
/F <JOB_NAME>,APPL=SHUT,?
```

Option	Short form	Comments
IMMEDIATE	IMM	Specifies that the Gateway daemon shuts down immediately. If you do not specify this option, a normal shutdown is performed.

Trace options

These options are available for use with the MVS system command `/F <JOB_NAME>,APPL=TRACE`.

To get help on these options, issue the following command:

```
/F <JOB_NAME>,APPL=TRACE,?
```

Option	Short form	Description
DUMPOFFSET =*integer*	OF=*integer*	Specifies the offset from which displays of any data blocks start, for example `512`. If the offset is greater than the total length of data to be displayed, an offset of 0 is used. This option applies only to the Gateway trace, not JNI trace. You cannot use this together with the `fulldatadump` option.
FULL DATA DUMP	FD	Sets the `dumpoffset` to 0 and ignores any value specified in `truncationsize`. This option applies only to the Gateway trace, not JNI trace.
		Specifies the name of the output file for JNI tracing. You must specify a value for this option. If you do not, an error is displayed. JNI trace is output as plain text, and there is no requirement to use a particular extension for the file name.
JNILEVEL=0|1	JL	**0** Off. No trace information is output. **1** On.
TFILESIZE =*integer*	TS=*integer*	Specifies the maximum size, in kilobytes, of the Gateway trace output file, for example `50000`.
		Specifies the output file for Gateway tracing, for example ./tracefile.trc on UNIX and Linux or .\tracefile.trc on Windows. If you do not specify a value for this option, trace output is sent to the console.
TLEVEL =1|2|3|4	TL	Specifies the Gateway trace level. Permitted values are: **0** Off. No trace information is output. **1** Exception tracing. Only exceptions are traced. See "Options on the ctgstart command" on page 254. **2** Trace exceptions, and entry and exit of methods. **3** Trace exceptions, some internals, and entry and exit of methods. **4** Full debug tracing (all trace points).

Option	Short form	Description
TRUNCATION SIZE =*integer*	TR=*integer*	Specifies the byte at which to stop the hex dump, for example `2000`. It defines the end point, not the number of bytes to display. So if on a dump of size 40 you set the `dumpoffset` to 11, and the `truncationsize` to 25, you will see 15 bytes (from 11 to 25). You cannot use this together with the `fulldatadump` option. This option applies only to the Gateway trace, not JNI trace.

Querying statistics

Options are available for selectively querying statistics.

To get help on these options, issue the following command:

```
/F <JOB_NAME>,APPL=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*.
RESOURCE GROUPS	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 315
You can use MVS system commands to display statistical information about the CICS Transaction Gateway, or obtain statistics using either the C or Java Statistics API interface.

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 MVS system command `/F <JOBNAME>,APPL=DUMP`.

To get help on the available dump options, issue the following command:

```
/F <JOB_NAME>,APPL=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	Generates a Java heap dump.
JAVA	Generates a Java dump.
JVM	Generates a dump containing current JVM memory usage.
JVMSTACK	Generates a dump containing only the Java call stack.
SYSTEM	Generates a Java system dump.

Responses

The Gateway daemon responds to a dump request with a message to the console. In some cases, if the request takes a long time to complete, the message NO RESPONSE RECEIVED is output, although the dump is produced. The stderr log contains the location of the dump.

IBM JVM dump responses

Messages from the JVM contain the name of the dump, and indicate whether the dump was successful. The JVM messages are sent to the stderr error stream.

Possible responses to the dump request are as follows:
- The Gateway daemon successfully processed the dump request; the dump request completed successfully.
- Null response received from the Gateway daemon during the dump request; the Gateway daemon received the dump request, but returned an invalid or null response.
- The dump type is unsupported in the remote JVM; the remote JVM does not support the requested dump type.
- An invalid response was returned from the Gateway daemon during the dump request; the Gateway daemon received the dump request, but the response returned was invalid.
- The Gateway daemon encountered a serious error while processing the dump type; the Gateway daemon received the dump request, but an error was detected.
- Some dump types are unsupported in the remote JVM; the remote JVM executing the Gateway daemon does not support some dump types.

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:

```
/F <JOB_NAME>,APPL=RMEXIT,?
```

Option	Short form	Description
*=command*COMMAND=*command*	*=command*CMD *=command*	The command that will be sent to all configured and active request monitoring user exits. This is a string. The eventFired() method is driven with a RequestEvent command. The *command* input data will be included as a string in the data map with RequestData key “CommandData”.

CICS request exit options

This option is available to interact with CICS request exits dynamically.

To get help on these options, issue the following command:

```
/F <JOB_NAME>,APPL=CREXIT,?
```

Option	Short form	Comments
COMMAND=*command*	CMD=*command*	The command that is sent to the CICS request exit. The value is a string. The eventFired() method is driven with an ExitEvent.Command event. The *command* input data is included as a string value in the data map for the key “ExitEventData.CommandData”.

Administering XA transactions with Resource Recovery Services

If communications between an application server and the CICS Transaction Gateway are interrupted and XA transactions are in progress, some transactions might be left in an *indoubt* state. To resolve these issues, restore communication with the application server, and use the application server system to correct the state of the transactions.

If communication with the application server cannot be restored, use Resource Recovery Services (RRS) under the control of the Interactive System Productivity Facility (ISPF) to resolve any indoubt transactions. WebSphere Application Server for z/OS manages resources for local mode transactions; shut it down before attempting to use RRS.

A copy of the XID is available in the RRS ISPF panel. For more information about RRS, see *Systems Programmers Guide to Resource Recovery Services (RRS), SG24-6980-00*.

The ISPF panels contain references to all the resource managers that are involved with a unit of recovery (UR). For XA requests, the name of the resource manager is based on the fully qualified APPLID of the Gateway daemon; it has a format of `CICSTG.`*`APPLIDQUALIFIER.APPLID`*. To identify all of the indoubt work from a gateway daemon instance, look for the work associated with the resource manager with the name `CICSTG.`*`APPLIDQUALIFIER.APPLID`* on the RRS panels.

If you attempt to manually back out CICS Transaction Gateway from RRS, and other resource managers are involved in a transaction, transactional integrity can be compromised.

Transactions that are initiated using the IPIC protocol are not viewable in RRS, and RRS configuration is not required for this protocol. For more information about IPIC queries on a CICS terminal, see "Administering transactions that use IPIC connections."

Administering transactions that use IPIC connections

Support for XA is provided for all transactions that use IPIC connections. You use the CEMT inquire command to discover the transactions that use IPIC connections.

To discover XA transactions that use IPIC connections, issue the following CEMT INQUIRE command from a CICS terminal:

```
CEMT INQUIRE UOWLINK IPIC
```

You can filter the transactions by entering an IP address or a wildcard for example:

```
CEMT INQUIRE UOWLINK IPIC HOST(9.20.*)
```

If the unit of work is part of an XA transaction (the default for all XA transactions using IPIC connections), the UOWLINK displays the global transaction identifier (GTRID) of the XID. Press PF2 in CEMT to find the hex value of GTRID. You can use this value to compare XIDs. Use the value to compare the CICS task to the corresponding transaction in the JEE application server.

When you have found your transaction, standard CICS management applies.

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 time outs are effected by this change:
- Gateway daemon close
- Gateway daemon idle
- Gateway daemon ping
- Server idle times
- Worker thread available thread

| **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.

Restarting Resource Recovery Services (RRS)

The CICS Transaction Gateway does not support a restart of the Resource Recovery Services (RRS) while any instances of the Gateway are running.

If RRS is restarted, a message is written to the JNI log. The following limitations apply to EXCI extended LUW and XA transactional requests.

- Non-transactional requests continue to work.
- The CICS Transaction Gateway must be restarted before transactional requests start working.

Chapter 12. Resolving problems

If a problem occurs you should first do some preliminary checks to try and narrow down the cause. You can then try and analyze the problem in more detail using tools such as trace, debug, or diagnostic commands.

A wide range of additional resources are also available for problem solving, including: forums and newsgroups, IBM Technotes, IBM Developerworks, and IBM Redbooks. You can also contact your IBM Support organization as described on the support page at http://www.ibm.com/support/entry/portal/Overview/Software/Other_Software/CICS_Transaction_Gateway .

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 167. You can also use the "JCA resource adapter installation verification test (IVT)" on page 165 to test that the connection from WebSphere Application Server through CICS Transaction Gateway to CICS Transaction Server is working correctly.

If you are currently upgrading CICS Transaction Gateway, ensure that you are aware of all the changes that have been made for this release, and make sure you have made any necessary configuration changes. For more information, see Chapter 4, "Upgrading," on page 25.

What messages were produced about the problem?

CICS Transaction Gateway writes information, warning and error messages to the message logs (for more information, see "General information about messages" on page 294). 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 Configuring.

What service changes have been applied since the last successful run?

If you have applied a PTF, 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 PTF to ensure that the instructions were followed correctly. If the PTF was installed correctly, try uninstalling it and see if the problem still occurs.

Is the problem related to a particular client application?

If you can identify a client application that is always in the system when the problem occurs, check it for coding errors. If the client application has not yet run successfully, examine it carefully to see if you can find any errors. If you have made changes to the client application since it last ran successfully, examine the new or modified part of the application. Consider the functions of the client application that might not have been fully exercised before.

Is the problem related to system loading?

If the problem seems to be related to system loading, the system might be running near its maximum capacity, or it might be in need of tuning. Check that you have defined sufficient resources (for example, connection manager threads and worker threads). Typically, if you had not defined sufficient resources, you might find that the problem is related to the number of users of the application.

Does the problem occur at specific times of day?

If the problem occurs at specific times of day, it could be dependent on system loading. Typically, peak system loading is at mid-morning and mid-afternoon, so those are the times when load-dependent problems are most likely to happen. Use the CICS TG interval statistics to determine when peak loading occurs and the resource usage at the time; for more information, see “Statistics” on page 309.

Regular backup jobs or other system maintenance might also cause unexpected problems at specific times of day.

What to do next

If preliminary checks have revealed the cause of the problem, you should now be able to resolve it, possibly with the help of other information in the CICS Transaction Gateway information center. If you have not yet found the cause of the problem, you must start to investigate it in greater detail.

To investigate the problem in more detail, begin by deciding the best category for the problem, for example is the problem related to installation, configuration or performance? Then go to “Dealing with problems” on page 270 where you will see a list of problems organized into the various categories. Each topic covers a single problem, and provides details on the symptom, probable cause and action to take.

If the problem is not listed in the categories, you might need to use one of the “Problem determination tools” or you might need to refer to “Problem solving and support” on page 299.

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 “Tracing” on page 295.

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 262.

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. This command is available under UNIX System Services.
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)

Command	Purpose
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. This command is available under UNIX System Services.
nslookup	Display information on Domain Name System (DNS) name servers. This command is available under UNIX System Services.
ping	Verify connection to a remote computer or computers. The equivalent command for IPv6 is ping6. This command is available under UNIX System Services.
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)

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.

Startup and shutdown problems

Problems when starting and stopping CICS Transaction Gateway.

Gateway daemon not able to access load library

When starting the Gateway, a CEE3250C ABEND S806 message is issued if the Gateway daemon cannot access the CICS EXCI load library.

Symptom

The following message is generated when starting the Gateway:

```
CEE3250C The system or user abend S806 R=00000004 was issued. From entry
point MVS_CcicsInit at compile unit offset -FFFF8D84 at address 1AEB158C.
```

Probable cause

The Gateway daemon cannot access the CICS EXCI load library.

Action

Check the STEPLIB environment variable setting and modify it if necessary. For more information, see “Environment variables: local and remote mode” on page 100 for more information. Also check that the user ID is authorized to access the CICS EXCI load library.

CTGRRMS services fails to initialize

CICS Transaction Gateway might fail to start the CTGRRMS services if the CTGRRMS version is incompatible.

Symptom

Message CTG8659E with return code 1792 is issued.

If CICS Transaction Gateway V7.2 or V8.0 is used with a V6.1, V7.0, or V7.1 CTGRRMS, the following message is also written to the system log:

```
IEF170I 1 CTGRRMS  IEA995I SYMPTOM DUMP OUTPUT 269
SYSTEM COMPLETION CODE=0C4  REASON CODE=00000011
...
ACTIVE LOAD MODULE       ADDRESS=1AD00048 OFFSET=0000117C
 NAME=CTGINIT
```

Probable cause

The return code 1792 might indicate that SCTGLINK installed with a version V6.1, V7.1 or V7.2 of CICS Transaction Gateway is specified on the LNKLST and CICS Transaction Gateway Version 7.2 or 8.0 is used.

Action

Configure the LNKLST to refer to the SCTGLINK data set that is installed with the release of the Gateway daemon you are using. See “Starting CTGRRMS services” on page 128 for more information. When multiple releases of the CICS Transaction Gateway are installed on the same z/OS image, the LNKLST must refer to the SCTGLINK that is installed with the latest release of the CICS Transaction Gateway.

Address space fails to initialize

If CTGRRMS is not running on the z/OS image and ctgasi is used to call it, CTGRRMS might fail to start.

Symptom

The following messages are issued:

```
CTG6201I ctgasi - CTGRRMS Services Address Space Initiator.

CTG6200I ctgasi - CTG6200I (C) Copyright IBM Corporation 2005.  All rights
reserved.

CTG6237I ctgasi - starting up services address space.

CTG6216E ctgasi - ASCRE failed for CTGINIT, Post value = FFFFFFFF.

CTG6255E ctgasi - timed out waiting for address space to initialize.

CTG6240E ctgasi - the version of CTGINIT may be incompatible with ctgasi.
```

Probable cause

- CTGRRMS service is not starting because CTGINIT on the LNKLST is at the wrong version. CTGRRMS is trying to reuse an *LX* value that is corrupt. You can further diagnose this by checking the MVS System Log for an 052 abend from the CTGRRMS process. If this dump is taking place, probably with a reason code of 0512, (in R15 for the dump), the reserved LX value might be corrupt.
- The timeout and post value of FFFFFF might indicate that the version of SCTGLINK referred to in the LNKLST is for a CICS Transaction Gateway V6.1,

V7.0 or V7.1 and a ctgasi is for a V7.2 or later of the CICS Transaction Gateway. CTGRRMS is trying to reuse an *LX* value previously used for the CTGRRMS services and the *LX* value is in use by another process. This might be due to a previous CTGRRMS process not terminating completely, or completing but not releasing the *LX* value.

There are two reasons why the message CTG6255E might be returned:
- CTGRRMS service is not starting because CTGINIT on the LNKLST is at the wrong version.
- CTGRRMS is trying to reuse an LX value that is corrupt. You can further diagnose this by checking the MVS™ System Log for an 052 abend from the CTGRRMS process. If this dump is taking place, probably with a reason code of 0512, (in R15 for the dump), the reserved LX value might be corrupt.

Action

1. Change the LNKLST to refer to a SCTGLINK PDS from the same version of the Gateway as the ctgasi tool. When multiple releases of CICS Transaction Gateway are installed on the same LPAR, SCTGLINK and ctgasi from the latest installed version must be used.
2. First check there is no CTGRRMS process running. If there is, force the process to end and retry. If the problem persists you can renew the *LX* value by starting CTGRRMS again using the force option: **`ctgasi -f`**.

   This removes the old LX value and a new value is chosen. Because a limited number of LX values available on a z/OS image, do not use this option unless there is no alternative. When all the LX values have been used you must IPL the z/OS image before new services, including CTGRRMS services, can be started.

CICS Transaction Gateway fails to initialize after an ARM restart

If CICS Transaction Gateway stops and if ARM (z/OS Automatic Restart Manager) is configured for cross-system restarts, it attempts to restart CICS Transaction Gateway on a different z/OS image within in the same sysplex.

Symptom

CICS Transaction Gateway fails to initialize following an Automatic Restart Manager (ARM) restart.

Probable cause

If z/OS fails, a whole group of related subsystems and applications also fail. ARM can restart all the failed systems automatically, in a predefined order, on a different z/OS image within the sysplex. This is called a *cross-system restart*.

In this situation, z/OS has failed and ARM has performed a cross-system restart of CICS Transaction Gateway on a different z/OS image in the same sysplex.

Action

Modify the ARM profile for the ARM_ID to prevent cross-system restarts.

For more information, see *z/OS V1R4.0 MVS Setting Up a Sysplex (SA22-7625-06)* at: `http://www-01.ibm.com/support/docview.wss?uid=pub1sa22762505`

CTGRRMS fails to start when XA support is active

If XA support has been activated, the Gateway daemon and CTGRRMS might not start.

Symptom

The CICS Transaction Gateway fails to start, and a log message indicates that CTGRRMS services could not be started.

Probable cause

The CTGINIT module is missing from the LNKLST.

Action

Check that the CTGINIT module is in the LNKLST. Dump symptoms, similar to the following, indicate that the CTGINIT module cannot be found:

```
CSV003I REQUESTED MODULE CTGINIT  NOT FOUND
CSV028I ABEND806-04  JOBNAME=MSTJCL00  STEPNAME=LLA
IEA989I SLIP TRAP ID=X806 MATCHED.  JOBNAME=CTGRRMS , ASID=018B.
IEE824I CTGRRMS  FAILED, TERMINATED
IEA995I SYMPTOM DUMP OUTPUT 495
SYSTEM COMPLETION CODE=806  REASON CODE=00000004
 TIME=12.33.27  SEQ=62869  CPU=0000  ASID=018B
 PSW AT TIME OF ERROR  070C1000   8142E1EE  ILC 2  INTC 0D
   NO ACTIVE MODULE FOUND
   NAME=UNKNOWN
   DATA AT PSW  0142E1E8 - 9024181E  0A0D18FB  180C181D
   GR 0: 00001E00   1: 84806000
      2: 00FCB218   3: 00000000
      4: 00000000   5: 008FD5C0
      6: 000000FF   7: 00000000
      8: 008FA250   9: 0142E6B4
      A: 00000000   B: 00000004
      C: 00000000   D: 008FA250
      E: 84806000   F: 00000004
 END OF SYMPTOM DUMP
```

Follow the steps in “Enabling CTGRRMS services” on page 127 to add the module to the LNKLST.

If a network failure occurs, the transaction is rolled back if the Java Transaction API (JTA) specification allows this. If the state of the transaction does not allow the transaction to be rolled back, the transaction manager repeatedly attempts to open new connections and reissue requests. The transaction manager rolls back the transaction or completes the transaction, depending on the state when the error occurs. In this situation, the transaction manager is responsible for rolling back or completing the transaction.

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 API requests, such as 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 Information Centers Library at: `http://www-01.ibm.com/software/htp/cics/library/`.

If normal shutdown fails you can promote this to an immediate shutdown.

CICS connection problems

Problems with connections to CICS Transaction Server.

Port is in use by another task

A permission denied error occurs if a port that CICS Transaction Gateway attempts to use is already in use by another task.

Symptom

The following error message is displayed:

```
EDC5111I PERMISSION DENIED
```

Probable cause

The port that CICS Transaction Gateway is attempting to use is already in use.

Action

Do one of the following:

- Enforce the exclusive use of the port for this CICS Transaction Gateway job. You can reserve a port for a given jobname in TCPIP.PROFILE. Reconfigure the address space that is already using the required port so that it uses a different port, then restart.
- Configure the Gateway daemon to use a different port number.

Conflict exists with a default port

A conflict exists between a default port used by CICS Transaction Gateway for one of its supported protocols and another port that is already in use.

Symptom

One or more protocols fail to start successfully and the following message is generated:

```
CTG6525E Unable to start handler for the protocol:protocol, port:port,
because:[RC]
```

Possible cause

A conflict might exist between the CICS Transaction Gateway default port and a port that is already in use.

Action

Change the port number in the configuration file (ctg.ini). For more information, see Configuring a remote mode topology.

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 88.

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.

EXCI connection problems

Problems when connecting to CICS over EXCI

EXCI pipe limit exceeded:

A problem with error ECI_ERR_SYSTEM_ERROR can occur if the EXCI pipe limit is exceeded during communication between CICS servers. This is due to an MVS system pipe limit.

Symptom

An ECI application has received the following return code:

```
-9 (ECI_ERR_SYSTEM_ERROR)
```

Probable cause

The EXCI pipe limit has been exceeded during communication between CICS servers.

A single z/OS address space is limited by CICS interregion communication (IRC) to allocating a maximum number of EXCI pipes for all attached CICS servers.

When the maximum limit has been reached, the next EXCI Allocate_Pipe call, made from a particular address space, fails with a SYSTEM_ERROR response code and a reason code 608. The ECI application receives a return code -9 (ECI_ERR_SYSTEM_ERROR), indicating that the maximum pipe limit has been exceeded.

Action

Modify the CICS system initialization parameter LOGONLIM to change the limit when MVS is initially loaded. You can allocate up to 250 pipes in an EXCI address space. The default limit is 100 pipes.

Pipe limit exceeded for available sessions:

A failure with error CTG6882E can occur if CICS Transaction Gateway tries to allocate more pipes than the number of available sessions defined in the CICS sessions definition. This is due to a CICS Transaction Gateway system pipe limit.

Symptom

The ECI application receives a return code -16 (ECI_ERR_RESOURCE_SHORTAGE) and DFHXCURM is invoked.

A RETRYABLE response code, and a reason code 202.

If the pipe limit is exceeded, a message is written to the JNI log, indicating the total number of pipes that the CICS Transaction Gateway was using at the time:

This tells you how many the CICS Transaction Gateway was using, and might be useful if other products are using pipes in the same address space.

Probable cause

The CICS Transaction Gateway has tried to allocate more pipes than there are available sessions defined in the CICS sessions definition, and the EXCI Open_Pipe call has failed.

Action

Consider setting the RECEIVECOUNT parameter in the CICS sessions to at least the maximum EXCI pipe limit. For more information, see "CICS server sessions definition" on page 123.

IPIC connection problems

Problems when connecting to CICS over IPIC

Unable to acquire IPCONN:

A problem can occur if the system cannot acquire an IPCONN.

Symptom

The following message appears in the CICS Transaction Server log, where *nnnn* is the CICS server name:

```
DFHIS1011 CICS_server_name Unable to acquire IPCONN IPCONN_name. An
EXCEPTION response to the capability exchange was received,
reason=SECURITY_VIOLATION
```

Probable cause

The IPCONN definition is not configured to use SSL and CICS Transaction Gateway, and the resource adapter is running on a different sysplex to CICS

Action

If you are using a resource adapter in local mode, configure it to use SSL. Put the Gateway on the same sysplex as the CICS server. For more information see

"Configurations that support identity propagation" on page 51.

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/`.

TCP/IP failure:

Problems with connections to CICS Transaction Server.

If a TCP/IP subsystem fails, the CICS Transaction Gateway tries to reconnect to TCP/IP when it becomes available again. However, if multiple TCP/IP stacks are in use on the z/OS system the CICS Transaction Gateway cannot detect that a TCP/IP subsystem has failed unless all the TCP/IP stacks to which it is bound fail. Consider using the TCP/IP environment variable _BPXK_SETIBMOPT_TRANSPORT to control which TCP/IP stacks the CICS Transaction Gateway can bind to; for more details see the *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Security problems

Problems with security.

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 Information Centers 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 z/OS system log:

`ICH408I USER` `*userid*` `GROUP` `*group*` `NAME` `*userid owner*` `DISTRIBUTED IDENTITY IS NOT DEFINED:` `*distinguished_name realm_name*`

Probable cause

RACF does not contain a mapping that associates the distinguished name of the user with a RACF user ID.

Action

If the user is permitted to access the CICS resources, create a RACF mapping that includes the distinguished name of this user. For more information see “Configuring RACF for identity propagation” in the CICS Transaction Server

Information Center at: `https://publib.boulder.ibm.com/infocenter/cicsts/v4r1/index.jsp`

Identity propagation login module not enabled:

The CICS Transaction Gateway identity propagation login module is not enabled and verification fails with an IRR012I message.

Symptom

The following message appears in the z/OS system log:

```
IRR012I VERIFICATION FAILED. USER PROFILE NOT FOUND
```

Probable cause

The CICS Transaction Gateway identity propagation login module is not enabled.

Action

Enable the CICS Transaction Gateway identity propagation login module in WebSphere Application Server.

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 96.

JSSE:

SSL problems with JSSE

Client not authorized to access key store:

The client application is denied access to the file system that contains the key store.

Symptom

The following Java exception occurs:

```
java.io.IOException: CTG6651E: Unable to connect to the Gateway.
                     [address = killerb2b, port = 8050]
                     [java.security.AccessControlException: access denied
                     (java.io.FilePermission \jssekeys\testclient.jks read)]
```

Probable cause

This happens if your application does not have permission to read from the file system containing the keystore.

Action

Configure SSL security correctly. For more information, see Configuring SSL.

Key ring file path not recognized:

The key ring file path is not recognized by Java.

Symptom

A Java exception occurs and a message similar to the following is generated:

```
CTG6525E Unable to start handler for the ssl: protocol because:
java.lang.Exception: java.io.IOException:R_datalib (IRRSDL00) error:
profile for ring not found (8, 8, 84)
```

Probable cause

A backslash (\) character has been incorrectly used as a directory separator in the path name of the key ring file and has not been recognized by Java.

Java interprets the backslash (\) character as a parameter delimiter. If you used the backslash character as a directory separator when entering the path name of the key ring file during configuration, Java cannot recognize the name.

Action

Modify the configuration path names to use either a forward slash (/) or double backslash (\\) as a separator in the path name on all operating systems.

For example:

```
/mykeys/jsse/keystore.jks
\\mykeys\\jsse\\keystore.jks
```

Problem importing a client certificate into RACF:

Unable to import a client certificate into RACF.

Symptom

The following message is displayed in the **RACF - Add digital certificate** dialog:

```
The input data set does not contain a valid certificate.
```

Probable cause

The client certificate is in the wrong data format.

Action

If you export as Base64 the output will be in EBCDIC encoding so will require conversion to ISO8859-1 from IBM-1047 before it can be imported using the keytool command.

128-bit encryption problem:

An encryption error might occur if the cipher suite is not specified correctly in the CICS Transaction Gateway configuration file.

Symptom

CICS Transaction Gateway starts with the following Java exception:

```
CTG6525E Unable to start handler for the ssl: protocol because:
java.lang.IllegalArgumentException:CTG6495E No cipher suites available for
use by SSL connection
```

If more than one cipher suite is used but only one is valid, the following message is displayed for the valid cipher suite (the other is ignored):

```
CTG8401I The following ciphers are enabled: SSL_RSA_WITH_NULL_SHA >
```

Probable cause

The cipher suite is not specified correctly in the **`ciphersuites`** parameter in the CICS Transaction Gateway configuration file.

Action

Edit the CICS Transaction Gateway configuration file to specify the cipher suite correctly.

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 and CICS Transaction Gateway or the resource adapter is running on a different sysplex to CICS (if the Gateway daemon is running on z/OS).

Action

Put the Gateway daemon on the same sysplex as the CICS server.

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.

Security error due to surrogate checking problem

An ECI_ERR_SECURITY_ERROR -27 can occur if a user ID is not authorized as a surrogate for the user ID specified on the ECI request.

Symptom

An ECI_ERR_SECURITY_ERROR -27 security error is issued.

Probable cause

Surrogate checking has been enabled in the EXCI options table DFHXCOPT but the user ID under which the CICS Transaction Gateway is running is not authorized as a surrogate for the user ID specified on the ECI request. The SURROGCHK option in the DFHXCOPT table enables surrogate checking. The default is YES; see “Customizing EXCI options” on page 119. The method used by the CICS Transaction Gateway to authenticate user ID and password, when AUTH_USERID_PASSWORD is set, changed with Version 5.0. Previously surrogate user checking was not performed even if the SURROGAT option was set in the DFHXCOPT options table on CICS. This change causes ECI requests to fail with a -27 security error if surrogate user checking is enabled and the user ID under which the CICS Transaction Gateway is running is not authorized as a surrogate for the user ID specified on the ECI request.

Action

See the *CICS Transaction Server for z/OS CICS External Interfaces Guide* and *CICS Transaction Server for z/OS RACF Security Guide* for more information about surrogate user checking.

Security error due to RACF problem

An ECI_ERR_SECURITY_ERROR -27 can occur if RACF program control is not active for the CICS Transaction Gateway load library.

Symptom

An ECI_ERR_SECURITY_ERROR -27 security error occurs.

Probable cause

RACF program control is not active for the CICS Transaction Gateway load library SCTGLOAD, and the CICS Transaction Server for z/OS SDFHEXCI load library. RACF program control must be active for the CICS Transaction Gateway load library SCTGLOAD, and the CICS Transaction Server for z/OS SDFHEXCI load library.

Action

Activate RACF program control:

```
SETROPTS CLASSACT(PROGRAM)
RDEFINE PROGRAM * UACC(READ)
SETROPTS WHEN(PROGRAM)
```

To add the CICS library when program control is active:

```
RALTER PROGRAM * ADDMEM('hlq.SDFHEXCI'/volser/NOPADCHK)
SETROPTS WHEN(PROGRAM) REFRESH
```

To add the CICS Transaction Gateway library when program control is active:

```
RALTER PROGRAM * ADDMEM('hlq.SCTGLOAD'/volser/NOPADCHK)
SETROPTS WHEN(PROGRAM) REFRESH
```

Additional information

Extended attributes settings are incorrect for certain HFS files.

Extended attributes for HFS files of the <install_path>/bin directory are set during the SMP/E installation process. However, if they are subsequently modified, program control might be compromised. Use the `ls -E` command from the USS shell command line to verify that extended attributes are set correctly.

The following `extattr` commands mark the load modules used by the CICS Transaction Gateway as program controlled. Issue commands similar to the following from an OMVS shell or a Telnet session:

```
extattr +p <install_path>/bin/lib*.so
extattr +ps <install_path>/bin/ctgstart
```

The Java SDK must also be program controlled. By default, the SDK is installed as program controlled. If necessary issue the following command:

```
extattr +p javapath/bin/*
```

where *javapath* is the location of the JVM. For further information, see “Configuring for client certificate mapping” on page 143.

Running ctgstart from the USS command line with AUTH_USER_PASSWORD set

To perform the necessary security calls to verify passwords, the Gateway daemon must run in a program controlled address space. Under the USS shell, the first non-program controlled program that runs (for example `ls`) makes that particular USS address space "dirty", and unable to subsequently run program controlled code.

Therefore, if you intend to run the Gateway daemon by executing the ctgstart script directly from a USS shell, set environment variable _BPX_SHAREAS to NO. This ensures that the Gateway daemon runs in a separate "clean" address space.

Note: This is in direct contrast to the setting of _BPX_SHAREAS used when executing the Gateway daemon via CTGBATCH.

The CICS Transaction Gateway failed to authenticate the user ID and password specified in the ECI call.

If user IDs and passwords are not to be authenticated within the CICS Transaction Gateway, ensure the variable AUTH_USERID_PASSWORD is not set in the CICS Transaction Gateway STDENV file or shell environment.

The JAVA_PROPAGATE environment variable has not been set for a CICS Transaction Gateway application running in local mode. You must set:

```
JAVA_PROPAGATE=NO
```

in the environment under which the application runs.

If the environment variable is not set, z/OS traces show that a pthread_security_np call with the CREATE_SECURITY_ENV parameter has failed with a 157 (EMVSERR) return code.

Program control error with security enabled

An error can occur if security is enabled and a load module is not program controlled.

Symptom

The following message is issued:

```
CTG6876E EXCI error: Function Call = function, Response = response, EXCI
Reason = return code, Subreason field-1 = return code, subreason field-2 =
return code, ctg_rc=error
```

with EXCI Reason=631.

You will also see system log entries similar to this:

```
ICH422I THE ENVIRONMENT CANNOT BECOME UNCONTROLLED.

CSV042I REQUESTED MODULE MODULE NOT ACCESSED. THE MODULE IS NOT PROGRAM
CONTROLLED

BPXP014I ENVIRONMENT MUST REMAIN CONTROLLED FOR DAEMON (BPX.DAEMON)
PROCESSING.
```

Probable cause

Security is enabled (AUTH_USERID_PWD is set to YES) and the load module *MODULE* is not program controlled. The system log messages are repeated if more than one load module is affected.

Action

Ensure that the load module is program controlled. For information about which load modules need to be program controlled if security is enabled see "Configuring for client certificate mapping" on page 143.

Security violation

An ECI_ERR_SYSTEM_ERROR return code might be generated as a result of problems in the CICS address space because of a security violation.

Symptom

An ECI_ERR_SYSTEM_ERROR return code.

Probable cause

A security violation has occurred.

Action

Check the MVS system log for more information about the cause of the error.

User not authorized to access DFHAPPL profiles

A problem might occur in the CICS address space because a user is not authorized to access DFHAPPL resources.

Symptom

An ECI_ERR_SYSTEM_ERROR return code and the following message:

```
DFH FC0400 applid This CICS system is not authorized to provide shared
access to data tables - reason code X'code'.
```

Probable cause

This error is generated in the EXCI code if the user ID of the CICS Transaction Gateway host address space is not permitted to the DFHAPPL.*dfhjvpipe* or the DFHAPPL.*applid* facilities.

CICS tried to open a data table but the table could not be shared with other CICS systems because a security check for update access to the resource name *DFHAPPL.applid* failed. The value of the reason code, *X'code'*, provides further information on the reason for the failure of the security check.

For more information see the CICS messages information in: `https://publib.boulder.ibm.com/infocenter/cicsts/v4r1/index.jsp`

Action

Check the MVS system log for more information about the cause of the error.

Memory problems

Problems caused by insufficient memory being available.

Memory use increases over time

The amount of memory used by the Gateway daemon might increases over time and a java.lang.OutOfMemory exception might occur.

The maximum number of connection manager threads and worker threads is defined in the CICS Transaction Gateway configuration.

Symptom

The Gateway daemon stops responding and the JVM writes a java.lang.OutOfMemory exception to the stderror log file or to the Java dump file. The JVM also creates various dump files in the information log. There is probably no noticeable decrease in performance before the problem occurs. If you happened to be monitoring memory usage before the dump occurred, you would have seen that memory usage gradually increased over time until eventually the limit was reached.

Probable cause

- There is a problem with a user-written application, for example a request exit which has remained inadvertently connected and is using Java resources.
- There are too many active Java threads (connection manager threads and worker threads).
- The Java heap size is unnecessarily large. Because the memory required to create Java heap and Java threads is allocated from the same finite storage area, it is possible that making the Java heap too large could indirectly cause a java.lang.OutOfMemory exception because there would then be insufficient memory available to create enough Java threads.
- The Java heap size is too small.
- The CICS REGION 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 60. For more information on Java memory allocation and JVM stack sizes, see the *IBM Java Diagnostics Guides Information Center*.

Also see “Tuning the gateway to avoid out of memory conditions” on page 61.

Related reference:

"List of statistics" on page 319
These statistics are available from the CICS Transaction Gateway.

Related information:

"Displaying statistics" on page 315
You can use MVS system commands 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 83
Edit the GATEWAY section of the configuration file to configure the Gateway daemon resources.

Resource problems

Problems due to shortage of resources.

Shortage of EXCI resources on the CICS server

An error can occur if there is a shortage of CICS receive sessions.

Symptom

Intermittent ECI_ERR_RESOURCE_SHORTAGE messages when sending an ECI request to CICS.

Probable cause

There are not enough receive sessions in the sessions definition.

Action

Increase the RECEIVECOUNT value on the SESSIONS resource definition on the CICS server to a value substantially greater than is theoretically necessary. Do this by setting RECEIVECOUNT to a value greater than the LOGONLIM value. For more information, see "CICS server sessions definition" on page 123.

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 111

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 262.

Failure to handle a Java exception

If an application or CICS Transaction Gateway fails to handle a Java exception, the Java Virtual Machine (JVM) writes a Java stack dump. This applies to Java exceptions in general.

Symptom

The JVM has written a Java stack dump.

Probable cause

An application or CICS Transaction Gateway failed to handle an exception.

Action

The destination for the dump output depends on your JVM implementation; check your Java documentation for more information.

To increase the information written to the Java stack dump, disable the Just-In-Time (JIT) compiler. The information included in the dump might include the line of Java source code where the exception happened. How you disable the JIT compiler depends on your JVM implementation; check your Java documentation for more information.

The following example shows a Java stack dump that was created with the JIT compiler disabled:

```
Exception in thread "main" java.lang.OutOfMemoryError
        at java.lang.Thread.start(Native Method)
        at com.ibm.ctg.server.ThreadManager.createObject
           (ThreadManager.java:345)
        at com.ibm.ctg.server.ThreadManager.<init>(ThreadManager.java:131)
        at com.ibm.ctg.server.ManagedResources.<init>
           (ManagedResources.java:106)
        at  com.ibm.ctg.server.JGate.main(JGate.java:895)
```

If the CICS Transaction Gateway handles an exception, a Java stack dump is written only if tracing is enabled. Try to reproduce the problem with tracing enabled because it helps to show you what was happening before the exception occurred. For more information, see Tracing.

Java class cache problems

Gateway daemon reports Java errors during initialization and, when -Xshareclasses option `nonfatal` is not specified, fails to start.

Symptom

Some, or all, of the following messages are written to the CICS TG job log:

```
JVMSHRC022E Error creating shared memory region
JVMSHRC336E Port layer error code = -302 or -308
JVMSHRC337E Platform error message: EDC5111I Permission denied.
JVMSHRC028E Permission Denied

ICH408I USER(userid ) GROUP(group-name ) NAME(user-name)
/tmp/javasharedresources/C240D2A32_semaphore_cicstgvrmgroup-name_Gnn
CL(FSOBJ ) FID(0000000000000000000000D200000000)
INSUFFICIENT AUTHORITY TO OPEN
ACCESS INTENT(RW-) ACCESS ALLOWED(GROUP R--)
EFFECTIVE UID(uid) EFFECTIVE GID(gid)

ICH408I USER(userid ) GROUP(group-name ) NAME(user-name)
/tmp/javasharedresources/C240D2A32_memory_cicstgvrmgroup-name_Gnn
CL(FSOBJ ) FID(0000000000000000000000D300000000)
INSUFFICIENT AUTHORITY TO OPEN
ACCESS INTENT(RW-) ACCESS ALLOWED(GROUP R--)
EFFECTIVE UID(uid) EFFECTIVE GID(gid)
```

Probable cause

CICS Transaction Gateway uses Java class caching and the user ID starting the Gateway daemon should have the relevant permissions to allow the user ID to create, or use existing, Java shared resources in the /tmp/javasharedresources directory. If the user ID starting the Gateway daemon does not have the relevant permissions some, or all, of the messages above are written to the CICS TG job log. CICS Transaction Gateway specifies the -Xshareclasses option `nonfatal`, which ensures that failure to access the Java shared class cache resources does not prevent the Gateway daemon from starting. If the -Xshareclasses option `nonfatal` is removed, the Gateway daemon will fail to initialize if it cannot access the Java shared class cache resources.

This might occur if the Gateway daemon is started by different user IDs that do not have the correct permissions to access the shared class cache resources in the /tmp/javasharedresources directory.

Action

Ensure that all user IDs starting a Gateway daemon have the correct permissions to access the tmp/javasharedresources directory and the Java shared class cache resources. See “Configuring Java shared classes” on page 97 for details of how to configure CICS TG with Java class caching.

For more information see your Java documentation.

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.

Corrupted data when using channels and containers

Data corruption when using channels and containers can occur if an incorrect CCSID is specified.

Symptom

Unexpected or corrupt data is returned to the client application when using an IPIC connection and channels and containers.

Probable cause

- The wrong CCSID is specified on the client application channel and has been inherited by the container.
- The wrong CCSID is specified on the container.

Action

1. If corrupted or unexpected data is returned, run a Gateway daemon trace to find out which code page the JVM is running on. Look in the System Properties section at the top of the trace.
2. For Java applications, use the setCCSID method to set the required code page on the channel. You must explicitly specify a CCSID when creating the container. For C or .NET applications, specify a CCSID when creating a CHAR container.

For more information on how to find the code page that the Client has sent to the server, see Data conversion.

WebSphere Application Server problems

Problems with WebSphere Application Server.

Authorization failure using servlets with WebSphere

A user ID and password authorization failure can occur for servlets that use the WebSphere autostart function. This is indicated by a message in the JNI trace.

Symptom

The following message appears in the JNI trace:

```
CcicsECI: Authorize userid/password with RACF. Return code = -1, errno =
157
```

Probable cause

Servlets that use the WebSphere autostart function do not have the required authority to access CICS Transaction Gateway.

Action

Use “user-driven” servlets. With these the initialization is performed in the servlet's init() method.

Gateway daemon startup fails due to dirty address space

The Gateway daemon fails to start because the address space where it runs has previously been used for running non program-controlled code.

Symptom

The Gateway daemon fails to start with the following error:

```
CTG9549E Gateway daemon failed to start, see Gateway error log; the log
location is defined by configuration file file name
```

Probable cause

The CICS Transaction Gateway HFS (Hierarchical File Store) is mounted with NOSETUID.

Action

Ensure that HFS is mounted with the default of SETUID.

Loss of connection with unit of recovery in doubt

A network connection between WebSphere Application Server and the Gateway daemon is terminated while a UOR is in doubt. This results in an ATR229D write to operator with reply (WTOR) message.

Symptom

A WTOR message similar to the following is written to the system log:

```
0000 *NN ATR229D CANCEL DELAYED. REPLY WAIT, BACKOUT, OR COMMIT TO RESOLVE
INDOUBT UR. URID = UR identifier.
```

Where *NN* is a number for the operator to reply to and *UR identifier* is a URID.

Probable cause

A network connection between WebSphere Application Server and the Gateway daemon is terminated, when a UOR is in doubt.

Action

No further action is required because WebSphere Application Server reconnects automatically and resolves the transaction.

When WebSphere Application Server reconnects to the Gateway daemon, or to any other Gateway daemon in the group, and issues a commit or backout request, the transaction is resolved, the WTOR is cancelled, and the following message is written to the log:

```
10:45:09.23 00000010 IEE400I THESE MESSAGES CANCELLED - NN
```

The connection manager thread that was dealing with the network connection that failed stays blocked until the UOR is resolved and the message cancelled.

Note: If you do reply to the message with BACKOUT or COMMIT before WebSphere Application Server has reconnected and resolved the transaction, a heuristic hazard error is returned to WebSphere Application Server. In this case it is your responsibility to ensure that transaction integrity is maintained. You should not respond to this message but instead, allow WebSphere Application Server to resolve the transaction.

Version problems

Problems due to product version non-compatibility.

Version compatibility problem with CTGRRMS

If CICS Transaction Gateway and the active CTGRRMS PC services have different versions, CICS Transaction Gateway terminates with a CTG9215E error.

Symptom

CICS Transaction Gateway terminates and the following error message is issued:

```
CTG9215E CICS TG version different from the active CTGRRMS PC services
```

Probable cause

The version of the CICS Transaction Gateway that you are using is not the same as the version of the CTGRRMS service address space that is running on the z/OS image. The executable code for CTGRRMS is supplied in CTGINIT which is shipped with CICS Transaction Gateway. If the version of CTGINIT is incremented at a new release of CICS Transaction Gateway, you must refresh CTGRRMS to start using the new version.

Action

Ensure that the versions of the CICS Transaction Gateway and the CTGRRMS services are the same. See “Refreshing CTGRRMS services” on page 129 for information on how to do this.

General information about messages

Information about message locations, formats, redirection, codes, and prefixes.

Message locations

The Gateway daemon writes log messages to standard output (stdout, referred to as the error log), or to standard error (stderr, referred to as the information log) . If Gateway daemon tracing is enabled, the Gateway daemon also writes log messages to the trace output file.

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	stdout
E	error	stderr
W	warning	stderr

Message redirection

Standard output and standard error can optionally be redirected to the same file. For more information, see “Starting in batch mode” on page 249.

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.

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.

Related reference:

“Gateway daemon logging” on page 87
Edit the GATEWAY section of the configuration file to configure the Gateway daemon logging resources.

Tracing

Tracing can be enabled and controlled for different components of the CICS Transaction Gateway.

Note: Tracing, especially debug tracing, decreases performance.

Related reference:

"Transaction tracking" on page 303
Transaction tracking can assist in diagnosing problems that sometimes occur when complex distributed transactions spread across a CICSplex.

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 MVS system commands see "Trace options" on page 261.

You can use options on the **ctgstart** command to enable and control tracing in the Gateway. For example, to enable standard trace, use the following command when starting the Gateway:

```
ctgstart -trace
```

For more information see "Options on the ctgstart command" on page 254.

Specifying trace output destination

If you do not set the TFILE parameter in the ctg.ini file, trace output is written to stderr. You can override the default destination for trace output by using the ctgstart -tfile option. For example, the command:

```
ctgstart -x -tfile=filename
```

starts the Gateway with debug tracing enabled and writes the trace output to the file specified by *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.

Tracing Java client applications

You can enable tracing in the application by using a Java directive when you start the JVM, or by adding calls to the CICS Transaction Gateway tracing API.

Use the **-D** option on the **java** command to specify Java directives. The tracing API comprises several static methods in the T class of the CICS Transaction Gateway. See the information about tracing in Java client programs in the *CICS Transaction Gateway for Multiplatforms: Programming Guide* for further information.

Tracing in Java Applets

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

Enable JNI trace by setting environment variables, by using a **`ctgstart`** command override, or when starting an application in local mode.

- While the CICS Transaction Gateway is running, use the MVS MODIFY command to enable JNI trace:
  ```
  /F JOB_NAME,APPL=TRACE,JNILEVEL=1
  ```
- When you start the CICS Transaction Gateway, issue the command:
  ```
  ctgstart -j-Dgateway.T.setJNITFile=filename
  ```

  where *`filename`* is the name of the file to which trace output is to be sent. If you do not specify a full path to the file, the location is <install_path>/bin.
- Set the following environment variables before you start the CICS Transaction Gateway or Java Client applications running in local mode:

  `CTG_JNI_TRACE`
  Use this environment variable to set the name of the JNI trace file. This environment variable only defines the name of the JNI trace file; it does not enable trace. JNI trace is output as plain text, and there is no requirement to use a particular extension for the file name.

  `CTG_JNI_TRACE_ON`
  Set this environment variable to YES (case-insensitive) to enable JNI trace when the CICS Transaction Gateway or Java Client application is started.
- For Java Client applications running in local mode, use Java to launch your application and set the system property gateway.T.setJNITFile, as shown in the following example:
  ```
  java -Dgateway.T.setJNITFile=filename application
  ```

  where
  - *`filename`* is the name of the file to which trace output is to be sent
  - *`application`* is the application to launch

When JNI tracing is enabled, some trace points might be written from threads running under different user IDs. To ensure that all trace points can be written to an existing file, change the permissions to allow all users to write to it:

```
chmod a+w filename
```

Do not allow more than one process to write to the same trace file, because this can cause trace points to be lost.

JEE tracing

A detailed trace mechanism is provided for the ECI resource adapter. Trace is useful when problem solving for applications that use the CICS resource adapters.

The CICS resource adapters support four levels of trace:

Level	Trace
0	No trace messages
1	Exception tracing only (default level)
2	Exception and method entry/exit trace messages
3	Exception, method entry/exit and debug trace messages

To provide more control over tracing, these system properties are available:

Property	Purpose
com.ibm.connector2.cics.tracelevel	Overrides the deployed trace level for the resource adapters without having to redeploy or deploy another CICS resource adapter.
com.ibm.connector2.cics.dumpoffset	The offset into a byte array at which a hex dump will start.
com.ibm.connector2.cics.dumplength	The maximum length of data displayed in a hex dump.
com.ibm.connector2.cics.outputerr	Declaring this directive sends trace output to standard error, if no other trace location has been specified either by the JEE server, or by the application developer working in a nonmanaged environment. In other circumstances the provided logwriter takes precedence.

These are JVM System properties that can be passed to the JVM on startup. The com.ibm.connector2.cics.tracelevel option is equivalent to the managed environment property "tracelevel" that is set as a custom property on the connection factory.

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\
```

This might map to:

```
/u/ctguser/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.

EXCI trace

The CICS Transaction Gateway writes trace entries to the EXCI trace. Entries are in the CICS trace EXCI format.

The trace entries in a dump can be printed using standard z/OS utilities.

To start an EXCI trace from the administration interface:

1. Use the command
   ```
   /D OMVS,A=ALL
   ```
   to display OMVS tasks.
2. Find the CICS Transaction Gateway task, and note the ASID.
3. Enter the DUMP command with a suitable comment. For example:
   ```
   /DUMP COMM=(JGATE DUMP)
   ```
4. Reply to the message with the ASID as follows:
   ```
   /R nn,ASID=aa,END
   ```
   where *nn* is the message number for the reply, and *aa* is the ASID.

See the System dumps section in the *CICS Transaction Server for z/OS CICS External Interfaces Guide* for information on how to format the trace.

Collecting SVC dumps of the Gateway daemon address space

The IBM support organization might request an SVC dump of the Gateway address space, or a problem might cause an SVC dump to be generated. Options are available for ensuring that information about the required components is captured.

Use the following command (or a similar command) to set the required dump options:

```
CD SET,SDUMP=(ALLPSA,SQA,SUMDUMP,NUC,RGN,LPA,TRT,CSA,GRSQ)
```

To take an SVC dump of a running Gateway daemon, use the following command sequence, setting the recommended options specifically for this dump instance:

```
DUMP COMM=(<Dump Title>)
R xx,SDATA=(<Dump Options>),CONT
R xx,JOBNAME=(<CICSTG Jobname>,OMVS),CONT
R xx,DSPNAME=('OMVS'.*),END
```

The command is shown split over several replies due to the restricted space available on the MVS system command line. You can enter it through the extended console command line as a single command:

```
DUMP COMM=(<Dump Title>),SDATA=(<Dump Options>),JOBNAME=(<CICSTG Jobname>,
OMVS),DSPNAME=('OMVS'.*)
```

The system log shows the result of the command, including the name of SVC dump data set, if successful.

Problem solving and support

This section provides information about how to resolve problems with your IBM software, including instructions for searching knowledge databases, downloading fixes, and getting support.

IBM Technotes and other support documents are published on the CICS Transaction Gateway support Web site. You can also search Web-based support resources by using the customized query fields in the Web search topic. For more information, see *http://www-01.ibm.com/software/htp/cics/ctg/support/*.

Searching knowledge bases

If you have a problem with CICS Transaction Gateway, you want it resolved quickly. Begin by searching the available knowledge bases to determine whether the solution to your problem is already documented.

1. Search the CICS Transaction Gateway Information Center.
2. Search the Internet. If you cannot find an answer to your question in the information center, search the Internet for the latest, most complete information that might help you resolve your problem. To search multiple Internet resources for CICS Transaction Gateway, use the Web search tool. The tool enables you to search a variety of resources including:
   IBM Technotes
   Downloads
   IBM Redbooks® publications
   IBM DeveloperWorks
   Forums and newsgroups
   Google

Contacting IBM Software Support

IBM Software Support provides assistance with product defects.

Before contacting IBM Software Support, your company must have an active IBM software maintenance contract, and you must be authorized to submit problems to IBM.

Follow the steps in this topic to contact IBM Software Support:

1. Determine the business impact of your problem.

2. Describe your problem and gather background information.
3. Submit your problem to IBM Software Support.

Determine the business impact of your problem

When you report a problem to IBM, you will be asked to supply a severity level. Therefore, you need to understand and assess the business impact of the problem you are reporting. Use the following criteria:

Severity	Impact	Characteristic
1	Critical	You are unable to use the program, resulting in a critical impact on operations. This condition requires an immediate solution.
2	Significant	The program is usable but is severely limited.
3	Moderate	The program is usable with less significant features (not critical to operations) unavailable.
4	Minimal	The problem causes little impact on operations, or a reasonable circumvention to the problem has been implemented.

Describe your problem and gather background information

When explaining a problem to IBM, be as specific as possible. Include all relevant background information so that IBM Software Support specialists can help you solve the problem efficiently. To save time, know the answers to these questions:

- What software versions were you running when the problem occurred?
- Do you have logs, traces, and messages that are related to the problem symptoms? IBM Software Support is likely to ask for this information.
- Can the problem be recreated? If so, what steps led to the failure?
- Have any changes been made to the system? For example, hardware, operating system, networking software, and so on.
- Are you currently using a workaround for this problem? If so, please be prepared to explain it when you report the problem.

To find out what information and files you will need to supply when opening a problem management record (PMR), see `http://www-01.ibm.com/support/docview.wss?uid=swg21287335#submit`

Submit your problem to IBM Software Support

You can submit your problem in one of two ways:

- Online: Go to the Submit and track problems page on the IBM Software Support site. Enter your information into the appropriate problem submission tool.
- By phone: For the phone number to call in your country, go to the contacts page of the IBM Software Support Handbook on the Web and click the name of your geographic region.

If the problem you submit is for a software defect or for missing or inaccurate documentation, IBM Software Support will create an Authorized Program Analysis Report (APAR). The APAR describes the problem in detail. Whenever possible, IBM Software Support will provide a workaround for you to implement until the APAR is resolved and a fix is delivered.

IBM publishes resolved APARs on the IBM product support Web pages daily, so that other users who experience the same problem can benefit from the same resolutions.

Chapter 13. Monitoring and statistics

Monitoring provides information about the status of individual requests. Statistics provide information about the performance of runtime components.

Monitoring

Request monitoring exits can optionally be used for driving user exit code on a per request basis. One or more user exit programs can be called for each request if details of each request are made available. All user exit code is called inline; this means that performance of the user exit code is critical.

Statistics

CICS Transaction Gateway statistics are always active and predefined by IBM. Unlike the request monitoring exits, the statistics provide summary information such as running totals, averages, status, and configuration values. Statistics are either displayed from system management commands, or can be obtained through a program that uses the statistics API. Statistics are optionally recorded to the System Management Facility (SMF) at regular intervals and are always written to SMF at the logical end of day.

Transaction tracking

Transaction tracking can assist in diagnosing problems that sometimes occur when complex distributed transactions spread across a CICSplex.

Transaction tracking across a CICSPlex

Transaction tracking can assist in diagnosing problems that sometimes occur when complex distributed transactions spread across a CICSplex.

Transaction tracking is available for Java, ECI V2 and .NET 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 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:

"Configuring identification using APPLID" on page 105
CICS Transaction Gateway supports identification using APPLID. This provides a standard mechanism for identification of Gateway daemon and Java client components in the CICSplex, and for subsequent task correlation in CICS.

"Client APPLID and APPLID qualifier" on page 108
Set the APPLID and APPLID qualifier for Client applications to enable transaction tracking.

Transaction tracking with Cross Component Trace (XCT)

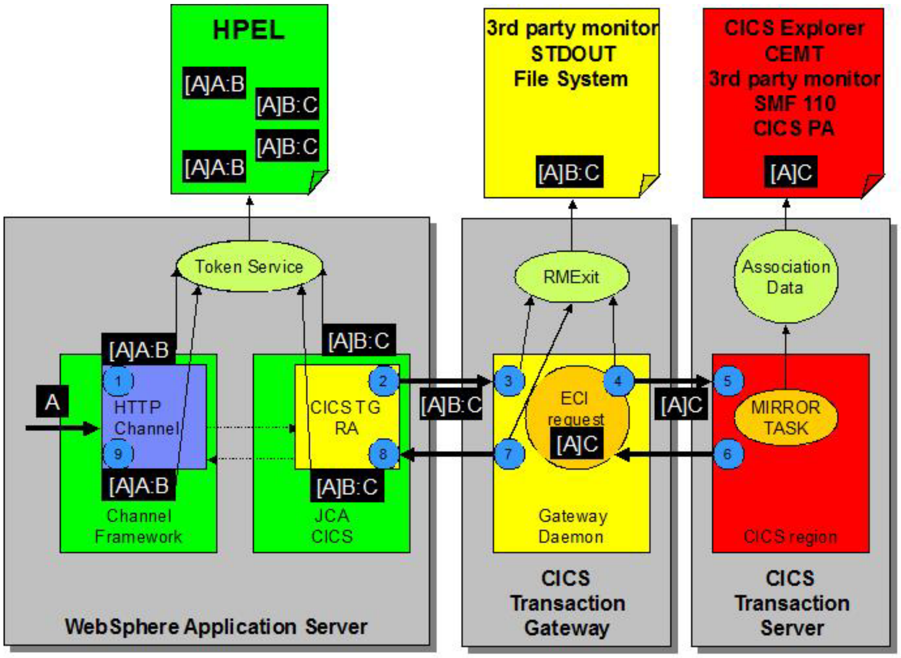
You can use Cross Component Trace (XCT) to track individual requests as they flow between WebSphere Application Server, CICS Transaction Gateway and CICS, assisting both problem diagnosis and system planning and configuration.

The XCT facility is available when using IPIC connections and 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 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.

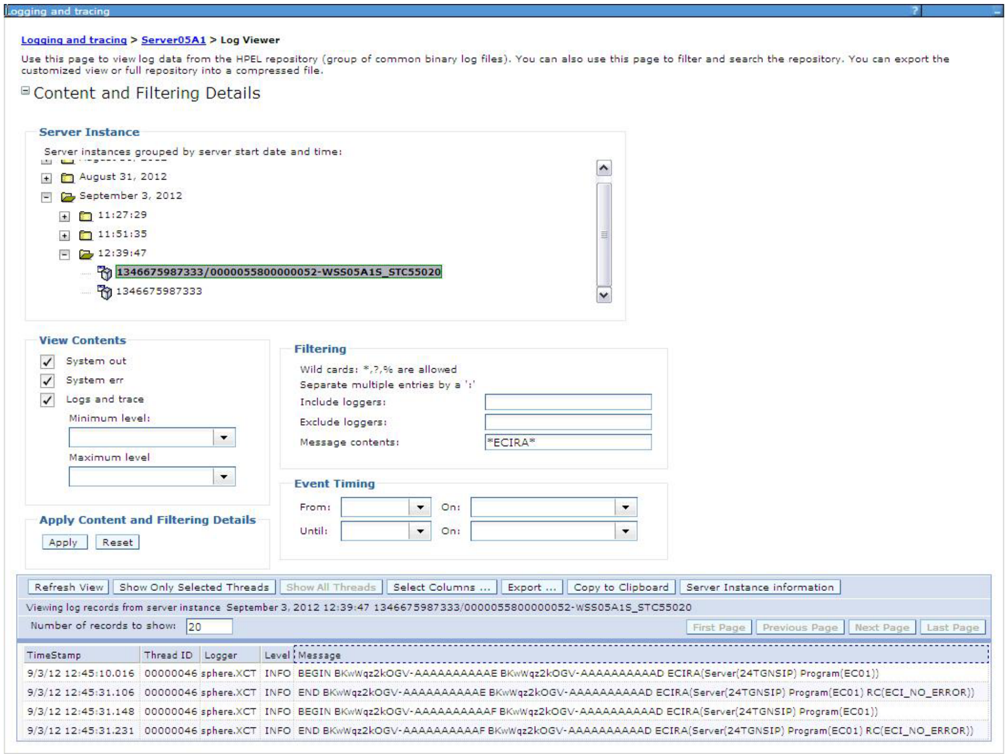


In the diagram above, A refers to the context assigned to the incoming request, B to the context of the Channel Framework, and C to the JCA CICS.

XCT data in 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 WebSphere Application Server Log Viewer:



Use this page to view log data from the HPEL repository (group of common binary log files). You can also use this page to filter and search the repository. You can export the customized view or full repository into a compressed file.

Server instances grouped by server start date and time:

Wild cards: *,?,% are allowed

Separate multiple entries by a ':'

Message contents: *ECIRA*

Viewing log records from server instance September 3, 2012 12:39:47 1346675987333/0000055800000052-WSS05A1S_STC55020

Number of records to show: 20

TimeStamp	Thread ID	Logger	Level	Message
9/3/12 12:45:10.016	00000046	sphere.XCT	INFO	BEGIN BKwWqz2kOGV-AAAAAAAAAAE BKwWqz2kOGV-AAAAAAAAAAD ECIRA(Server(24TGNSIP) Program(EC01))
9/3/12 12:45:31.106	00000046	sphere.XCT	INFO	END BKwWqz2kOGV-AAAAAAAAAAE BKwWqz2kOGV-AAAAAAAAAAD ECIRA(Server(24TGNSIP) Program(EC01) RC(ECI_NO_ERROR))
9/3/12 12:45:31.148	00000046	sphere.XCT	INFO	BEGIN BKwWqz2kOGV-AAAAAAAAAAF BKwWqz2kOGV-AAAAAAAAAAD ECIRA(Server(24TGNSIP) Program(EC01))
9/3/12 12:45:31.231	00000046	sphere.XCT	INFO	END BKwWqz2kOGV-AAAAAAAAAAF BKwWqz2kOGV-AAAAAAAAAAD ECIRA(Server(24TGNSIP) Program(EC01) RC(ECI_NO_ERROR))

The XCT context data can also be seen in the 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
         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
         thread=[WebSphere WLM Dispatch Thread t=007c40b8] requestID=[BKwWqX+HPuK-AAAAAAAAAAG]
         END   BKwWqz2kOGV-AAAAAAAAAAE BKwWqz2kOGV-AAAAAAAAAAD ECIRA(Server(24TGNSIP) Program(EC01)
```

Additional information can be viewed using the WebSphere Application Server Cross Component Trace.

For more information about configuring Cross Component Trace, see the WebSphere Application Server V8.5 information center"

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:

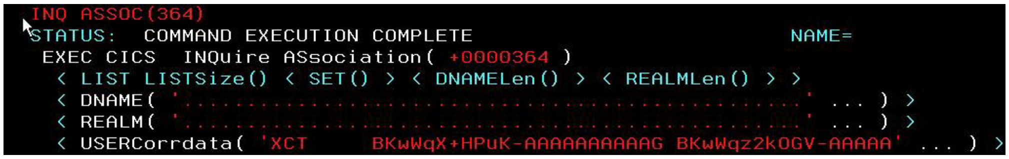
```
[00000000001]: com.ibm.ctg.samples.requestexit.BasicMonitor:eventFired called with event = Reques
  : : : : : :
[00000000001]: XctRoot = BKwWqX+HPuK-AAAAAAAAAAG
[00000000001]: XctParent = BKwWqz2kOGV-AAAAAAAAAAD
[00000000001]: XctCurrent = BKwWqz2kOGV-AAAAAAAAAAE

[00000000001]: com.ibm.ctg.samples.requestexit.BasicMonitor:eventFired called with event = Reques
 : : : : : :
[00000000001]: OriginData - Transaction Group ID = 1A10C2C1 E8D3C9E2 E22EC7C1 E3C5E6C1 E8F1CA1D E
                          - User Correlator = XCT     BKwWqX+HPuK-AAAAAAAAAAG BKwWqz2kOGV-AAAAAAA
 : : : : : :
[00000000001]: XctRoot = BKwWqX+HPuK-AAAAAAAAAAG
[00000000001]: XctParent = BKwWqz2kOGV-AAAAAAAAAAD
[00000000001]: XctCurrent = BKwWqz2kOGV-AAAAAAAAAAE

[00000000001]: com.ibm.ctg.samples.requestexit.BasicMonitor:eventFired called with event = Respor
 : : : : : :
[00000000001]: OriginData - Transaction Group ID = 1A10C2C1 E8D3C9E2 E22EC7C1 E3C5E6C1 E8F1CA1D E
                          - User Correlator = XCT     BKwWqX+HPuK-AAAAAAAAAAG BKwWqz2kOGV-AAAAAAA
 : : : : : :
[00000000001]: XctRoot = BKwWqX+HPuK-AAAAAAAAAAG
[00000000001]: XctParent = BKwWqz2kOGV-AAAAAAAAAAD
[00000000001]: XctCurrent = BKwWqz2kOGV-AAAAAAAAAAE
```

For more information about when the XCT request identifiers are available, see Data available by FlowType and RequestEvent Data available by FlowType and RequestEvent

XCT data in CICS

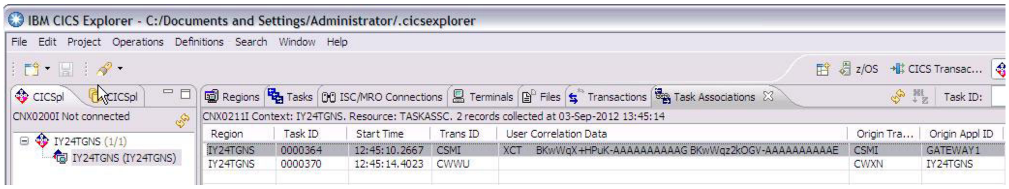
For IPIC only, the user correlation data containing the root and current XCT contexts is sent from CICS TG to CICS as part of the origin data. In addition to obtaining the user correlation data through the CICS API, the user correlation data can be viewed in the task's association data, using the command 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):



```
INQ ASSOC(364)
STATUS:  COMMAND EXECUTION COMPLETE                              NAME=
 EXEC CICS  INQuire ASsociation( +0000364 )
  < LIST LISTSize() < SET() > < DNAMELen() > < REALMLen() > >
  < DNAME( '.................................................' ... ) >
  < REALM( '.................................................' ... ) >
  < USERCorrdata( 'XCT     BKwWqX+HPuK-AAAAAAAAAAG BKwWqz2kOGV-AAAAA' ... ) >
```

The CICS Explorer can also be used to display the user correlation data, for example:



CNX0211I Context: IY24TGNS. Resource: TASKASSC. 2 records collected at 03-Sep-2012 13:45:14

Region	Task ID	Start Time	Trans ID	User Correlation Data	Origin Tra...	Origin Appl ID
IY24TGNS	0000364	12:45:10.2667	CSMI	XCT BKwWqX+HPuK-AAAAAAAAAAG BKwWqz2kOGV-AAAAAAAAAAE	CSMI	GATEWAY1
IY24TGNS	0000370	12:45:14.4023	CWWU		CWXN	IY24TGNS

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 22. Data available to request monitoring exits

Request data description	Description
Channel	Channel 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 generated by the Java client application.
ClientLocation	The location of the Client application (IP address).
CommandData	Command data originating 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 used to track this flow within the CICS Transaction Gateway instance.
CtgReturnCode	CICS Transaction Gateway return code on a response.
DistributedIdentity	Distributed identity associated with this transaction.
FlowTopology	Topology from which the request exit was called: • Gateway - from the Gateway daemon • RemoteClient - from a remote client • LocalClient - from a local client
FlowType	Flow type of this request or response.
GatewayUrl	URL of the Gateway to which the Java client is connecting.
Location	Location of this monitor. The value is an IP address.
LuwToken	CICS Transaction Gateway logical unit of work token.
OriginData	Data identifying the Client application that originated a CICS task and which contains the APPLID and APPLID qualifier. This 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.

Table 22. Data available to request monitoring exits (continued)

Request data description	Description
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.
Urid	RRMS URID for this XA transaction. This value is supported exclusively for XA transactions on z/OS and is an array of 16 bytes.
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 information:

Java request monitoring exits

Request monitoring exits configuration

Request monitoring exit API information

Request monitoring exits configuration

In a remote mode topology, you can configure request monitoring exits individually for the Gateway classes and the Gateway daemon. In a local mode topology, you 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 151.

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 from which they were collected. Statistical data is based on Client applications that run in remote mode. No statistical data is available for Java Client applications that run in local mode. The collection of statistics has an insignificant impact on performance, and statistics are always available.

CICS Transaction Gateway statistics aim to assist you in the following activities:
- Capacity planning information and throughput analysis
- Critical resource usage
- Problem determination

Interval and end-of-day statistics reflect those used by the CICS Transaction Server products to allow for synchronization of statistics collection between the products. Further information can be found from the Statistics parameters section of the *CICS Transaction Server for z/OS System Definition Guide*.

Resource group ID

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 319 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 319

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 SMF

CICS Transaction Gateway provides a mechanism to record interval, end-of-day data, and shutdown statistics data to the System Management Facility on z/OS. For more information, see “Recording statistics to SMF” on page 338.

Statistics configuration

You can configure parameters for statistics interval, statistics end of day, recording statistics to SMF, 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, all statistics are optionally recorded to SMF, and statistics of statistical type "Interval" are reset to the default values. For examples, see “Interval timing patterns” on page 313.

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 using the Gateway settings in the configuration file.

CICS writes the interval statistics to the SMF data set automatically at the expiry of the interval if the statistics recording status is set ON by the **statsrecording** system parameter.

The **statint** parameter equates to the CICS Transaction Server keyword and concepts of the parameter of the same name.

See “Statistics interval” on page 154 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. At this point, the interval ends and data is written to SMF regardless of the setting of the **statsrecording** system parameter.

You can change the end of day value using the Gateway settings in 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 154 for more information.

Recording to SMF

CICS Transaction Gateway provides a mechanism for recording interval statistics and end-of-day data to the System Management Facility (SMF) on z/OS.

You can set the **statsrecording** parameter using the Gateway settings in the configuration file.

See “Enable statistic recording to SMF” on page 155 for more information.

Statistics API port

The Statistics API port allows the Gateway daemon to handle incoming requests for the Statistics API. You can select the port number on which to listen for Statistics API requests using the Gateway settings in the configuration file.

See “Statistics API protocol settings” on page 152 for more information.

Related reference:

“Statistics interval” on page 154
The **`statint`** parameter specifies the recording interval for system statistics.

“Statistics end of day time” on page 154
The **`stateod`** parameter specifies the end of day time.

“Recording statistics to SMF” on page 338
CICS Transaction Server, TXSeries, and WebSphere support the recording of statistics to SMF. The z/OS based products write SMF records. CICS Transaction Gateway provides similar function on the z/OS platform using type 111 SMF record format.

“Statistics API protocol settings” on page 152
To configure the statistics API protocol settings, edit the statistics API protocol parameters in the GATEWAY section of the configuration file.

Setting up your system for statistics

To configure your system to deal with requests for statistics edit the configuration file, `ctg.ini`, following these steps.

Procedure

1. Add the **`statsapi`** handler to the GATEWAY section of the configuration file.
2. Set the statistics interval parameter **`statint`**. The parameter has the format `HHMMSS`. The default value is three hours (`030000`).
3. Set the end of day parameter **`stateod`**. The parameter has the format `HHMMSS`. The default value is midnight (`000000`).
4. Set the statistics recording parameter **`statsrecording`** to `on` to enable statistics to be recorded to SMF.
5. Save the configuration file, then stop and restart the Gateway daemon.

Related reference:

“Statistics API protocol settings” on page 152
To configure the statistics API protocol settings, edit the statistics API protocol parameters in the GATEWAY section of the configuration file.

Related information:

“Statistics API protocol parameters” on page 161
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 **Statistics Interval** combines with the **Statistics End of Day time** to formulate times at which interval statistics are reset. Interval statistics can also be optionally recorded. 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.

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 23. 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

Table 23. Interval Statistics timing – Example 1 (continued)

Time	Event type	Interval length HH:MM:SS
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 24. Interval Statistics timing – Example 2

Time	Event type	Interval length HH:MM:SS
05:20:00	Gateway starts	Not applicable
05:59:00	Interval reset	00:39:00
11:59:00	Interval reset	06:00:00
17:59:00	Interval reset	06:00:00
23:59:00	End of Day reset	06:00:00
05:59:00	Interval reset	06:00:00
11:59:00	Interval reset	06:00:00
Sequence repeats		

Interval Statistics timing - Example 3:

The Gateway daemon starts at 5:20 a.m. (052000) and is configured for a 24 hour statistics interval, with logical end of day at 23:59 with **`statint=240000 stateod=235900`**.

The first Interval is scheduled to end at 23:59 (235900), and is of 17 hours and 39 minutes duration. This schedule allows subsequent intervals be aligned with the end-of-day event. In this case, statistics are reset and optionally recorded at the following times during the days following Gateway initialization:

Table 25. 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 MVS system commands to display statistical information about the CICS Transaction Gateway, or obtain statistics using either the C or Java Statistics API interface.

Use MVS system commands 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 using MVS system commands:

```
/F <JOBNAME>,APPL=STATS,<OPTIONS>
```

For example, to display all available statistics about the CICS Transaction Gateway, enter the following command:

```
/F <JOBNAME>,APPL=STATS,GS
```

The command is not case-sensitive.

Displaying all available statistics

Use the administration interface with the GS option with no parameters, to display all available statistical information about the CICS Transaction Gateway.

Enter the following command:

```
/F <JOBNAME>,APPL=STATS,GS
```

Selecting the statistics to display

Use the administration interface , 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:

```
/F <JOBNAME>,APPL=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:

```
/F ,APPL=STATS,GS,ST=I
```

```
/F ,APPL=STATS,GETSTATS,STATTYPE=I
```

To display the current statistical values from the CS resource group, enter the following command:

```
/F ,APPL=STATS,GS=CS,ST=C
```

To display all lifetime and interval statistical values from the GD resource group, enter the following command:

```
/F ,APPL=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 the administration interface, together with the RG parameter with no other options, to list available resource groups.

To list available resource groups, enter the following command:

```
/F <JOBNAME>,APPL=STATS,RG
```

Listing all available statistical IDs

Use the administration interface, together with the SI parameter, to list available statistical IDs.

To list all available statistical IDs, enter the following command:

```
/F <JOBNAME>,APPL=STATS,SI
```

Listing statistical IDs for selected resource groups

Use the administration interface, 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:

```
/F <JOBNAME>,APPL=STATS,SI=CM:WT
```

Getting help on statistics

Use the administration interface with the **`?`** option to get help on statistics.

Issue the following command:

```
/F <JOBNAME>,APPL=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 the administration interface and the statistical API to retrieve statistics, and is not case-sensitive.

Name
The name of the resource group, displayed when the administration interface is used to display statistical information.

Description
A description of the resource group.

These resource groups are defined:

Table 26. Resource groups

ID	Name	Description
CM	Connection manager statistics	Statistics about connection manager threads.
CS	CICS server (all) statistics	Statistics about all CICS servers.
CS*x*	CICS server (instance) statistics	Statistics for an individual CICS server, where *x* is the APPLID of the CICS server.
GD	Gateway daemon statistics	Statistics on transaction counts, request counts, and Gateway status.
LS	Logical CICS server (all) statistics	Statistics for logical CICS servers.
LS*x*	Logical CICS server (instance) statistics	Statistics for an individual logical CICS server.
PH	Protocol handler statistics	Statistics about protocol handlers.
SE	System environment statistics	Statistics about the System Environment of the Gateway daemon.
WT	Worker thread statistics	Statistics about worker threads.

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 a CICS server connected over IPIC or EXCI 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.

Logical CICS server (all) resource group (LS)

CICS Transaction Gateway records the number of requests, for all associated logical CICS servers, that have been remapped, since startup and since the start of the interval. A runtime statistic enables the proportion of mapped requests to be calculated.

Logical CICS server (instance) resource group (LS*x)*

Statistics are available for each specific logical CICS server "x" to show the number of requests that have been received for this logical CICS server in the interval and since startup.

Protocol handler resource group (PH)

Statistics are available to identify whether the SSL and TCP protocol handlers are enabled and the port numbers in use. A 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. In addition, statistical values representing region storage availability and current usage are provide on the z/OS platform.

Worker thread resource group (WT)

Statistics are available for the worker threads. These statistics identify the characteristics for the pool of worker threads that can be used by connection managers. These statistics are useful for analyzing resource usage and capacity planning, and for diagnosing system problems.

List of statistics

These statistics are available from the CICS Transaction Gateway.

Each statistic has the following characteristics:

ID A unique identifier for the statistic. The ID is used by the administration interface 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 the administration interface is used to display statistical information.

Description
A description of the information returned by the statistic.

Value returned
The type of information returned by the statistics:

Integer
The string value represents a 4-byte numeric value.

Long The string value represents an 8-byte numeric value.

String The string value represents character data.

These subtopics describe the statistics that are defined:

Connection manager statistics:

The statistics listed here belong to the connection manager resource group.

Table 27. Connection manager statistics

ID	Description	Default value	Data type
CM_CALLOC	The current number of connection manager threads allocated to clients.	0	Integer
CM_CCURR	The current number of connection manager threads created.	0	Integer
CM_CWAITING	The current number of connection managers waiting for a worker thread to become available.	0	Integer
CM_IALLOC	The number of allocations for connection manager threads representing the number of connections that have been established from remote clients. A low value represents efficient connection reuse.	0	Integer
CM_IALLOCHI	The peak number of connection manager threads concurrently allocated to client applications. This number represents a high water mark for CM_CALLOC.	<CM_CALLOC>	Integer
CM_ICREATED	The number of connection manager threads created.	0	Integer
CM_ITIMEOUTS	The number of times that the Gateway daemon failed to allocate a connection manager thread to a client application within the defined **`connecttimeout`** length of time.	0	Integer
CM_LALLOC	The number of allocations for connection manager threads representing the number of connections that have been established from remote clients. A stable value represents efficient connection reuse.	0	Integer
CM_LTIMEOUTS	The number of times that the Gateway daemon failed to allocate a connection manager thread to a client application within the defined (connecttimeout) length of time.	0	Integer
CM_SINIT	The initial number of connection manager threads **`initconnect`** created by the Gateway daemon.	0	Integer
CM_SMAX	The maximum number of connection manager threads **`maxconnect`** that can possibly be created and allocated by the Gateway daemon.	0	Integer **Note:** A value of -1 indicates no limit.

CICS server (all) statistics:

The statistics listed here belong to the CICS server (all) resource group.

Table 28. CICS server (all) statistics

ID	Description	Default value	Data type
CS_CALLOC	The current number of allocated EXCI pipes across all CICS servers.	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_CWAITING	The number of requests currently waiting for a response from a CICS server.	0	Integer
CS_IALLOCFAIL	The number of times the Gateway daemon has tried and failed to allocate a pipe to a CICS server.	0	Integer
CS_IALLREQ	The number of requests to CICS servers (successful and failed) that have been processed.	0	Integer
CS_IAVRESP	The average time taken (in milliseconds) for a connected CICS server to respond to the Gateway daemon over the course of this interval.	0	Integer
CS_ICOMMSFAIL	The number of times communication with a CICS server 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_ICONNFAIL	The number of times an attempt to connect to a CICS server has failed.	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_IIDLETIMEOUT	The number of times a connection to a CICS server has timed out.	0	Integer
CS_ILIST	The list of CICS servers to which requests have been sent.	<empty string>	String
CS_ILOSTCONN	The number of times an established connection with a CICS server has been lost.	0	Integer
CS_IREALLOC	The number of times that a worker thread has to deallocate its existing pipe and reallocate a new pipe because the existing pipe points to a different CICS server from that required by the request. This statistic applies only when the environment variable CTG_PIPE_REUSE is set to ONE. Cases where the pipe for the worker thread was deallocated and reallocated because the CICS server was not available are excluded.	0	Integer

Table 28. CICS server (all) statistics (continued)

ID	Description	Default value	Data type
CS_IREQDATA	The amount of request data (in bytes) sent to connected CICS servers. This amount includes both application and CICS protocol data.	0	Long
CS_IRESPDATA	The amount of response data (in bytes) received from connected CICS servers. This amount includes both application and CICS protocol data.	0	Long
CS_ISESSFAIL	The number of failures on IPIC sessions to CICS servers.	0	Integer
CS_LALLOCFAIL	The number of times the Gateway daemon has tried and failed to allocate a pipe to a CICS server.	0	Integer
CS_LALLREQ	The number of requests to CICS servers (successful and failed) that have been processed.	0	Integer
CS_LAVRESP	The average time taken (in milliseconds) for a connected CICS server to respond to the Gateway daemon over the lifetime of the current Gateway daemon.	0	Integer
CS_LCOMMSFAIL	The number of times communication with a CICS server has failed after a connection has already been established, or when there has been a failure with the link during communication with the server. In EXCI, a pipe was available but the request failed.	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_LREALLOC	The number of times that a worker thread has to deallocate its existing pipe and reallocate a new pipe because the existing pipe points to a different CICS server from that required by the request.	0	Integer
CS_LRESPDATA	The amount of response data (in bytes) received from connected CICS servers. This amount includes both application and CICS protocol data. For IPIC, the data comprises COMMAREA and CICS headers.	0	Long

Table 28. CICS server (all) statistics (continued)

ID	Description	Default value	Data type
CS_LREQDATA	The amount of request data (in bytes) sent to connected CICS servers. This amount includes both application and CICS protocol data. For 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_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_SLOGONLIM	The maximum EXCI pipe allocation for each MVS address space. This value is the equivalent to the CICS subsystem initialization parameter LOGONLIM.	100	Integer
CS_SNETNAME	The NETNAME of the specific pipe used for EXCI calls or empty string if generic pipes are being used. The NETNAME is defined by the environment variable DFHJVPIPE.	<empty string>	String

CICS server (instance) statistics:

The statistics listed here belong to the CICS server (instance) resource group.

Table 29. CICS server (instance) statistics

ID	Description	Default value	Data type
CS*x*_CALLOC	The current number of allocated EXCI pipes for CICS server *x*.	0	Integer
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*_CSESSCURR	The number of IPIC sessions in use with CICS server *x*.	0	Integer
CS*x*_CSESSMAX	The number of IPIC sessions negotiated with CICS server *x*.	0	Integer
CS*x*_CWAITING	The number of requests currently waiting for a response from CICS server *x*.	0	Integer
CS*x*_IALLOCFAIL	The number of times the Gateway daemon has tried and failed to allocate a pipe to CICS server *x*.	0	Integer
CS*x*_IALLREQ	The number of requests to CICS server *x* (successful and failed) that have been processed.	0	Integer
CS*x*_IAVRESP	The average time taken (in milliseconds) for connected CICS server *x* to respond to the Gateway daemon.	0	Integer

Table 29. CICS server (instance) statistics (continued)

ID	**Description**	**Default value**	**Data type**
CS*x*_ICOMMSFAIL	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. In EXCI, a pipe was available but the request failed.	0	Integer
CS*x*_ICONNFAIL	The number of times an attempt to connect to a CICS server has failed.	0	Integer
CS*x*_IIDLETIMEOUT	The number of times a connection to CICS server *x* has timed out.	0	Integer
CS*x*_ILOSTCONN	The number of times an established connection with a CICS server has been lost.	0	Integer
CS*x*_IREQDATA	The amount of request data (in bytes) sent to connected CICS server *x*. This amount includes both application and CICS protocol data.	0	Long
CS*x*_IRESPDATA	The amount of response data (in bytes) received from connected CICS server *x*. This amount includes both application and CICS protocol data.	0	Long
CS*x*_ISESSFAIL	The number of failures on IPIC sessions to CICS server *x*.	0	Integer
CS*x*_LALLOCFAIL	The number of times the Gateway daemon has tried and failed to allocate a pipe 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. In EXCI, a pipe was available but the request failed.	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*_LREQDATA	The amount of request data (in bytes) sent to connected CICS server *x*. This amount includes both application and CICS protocol data. For IPIC, the data comprises COMMAREA and CICS headers.	0	Long

Table 29. CICS server (instance) statistics (continued)

ID	Description	Default value	Data type
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 IPIC, the data comprises COMMAREA and CICS headers.	0	Long
CS*x*_LSESSFAIL	The number of failures on IPIC sessions to CICS server *x*.	0	Integer
CS*x*_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*_SPROTOCOL	The protocol used to communicate with the CICS server *x*. The protocol name is one of the following: EXCI, IPIC, IPICSSL, SNA, TCPIP.	N/A	String
CS*x*_SSESSMAX	The number of requested IPIC sessions for CICS server *x*.	0	Integer

Gateway daemon statistics:

The statistics listed here are members of the Gateway daemon resource group.

Table 30. Gateway daemon statistics

ID	Description	Default value	Data type
GD_CHEALTH	The current health of communications between the Gateway daemon and CICS.	100	Integer
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_CXATXN	The current number of inflight XA 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
GD_IALLREQ	The number of API calls (ECI, ESI) and XA requests that have been processed. Successful and failed requests are included. Administrative requests and handshakes are excluded.	0	Integer

Table 30. Gateway daemon statistics (continued)

ID	Description	Default value	Data type
GD_IAVRESP	The average time taken in milliseconds for the Gateway daemon to respond to API (ECI, ESI) and XA requests from remote clients. Successful and failed requests are included. This value is inclusive of the CICS response time, as provided by the corresponding CS_IAVRESP statistic.	0	Integer
GD_IAVRESPIO	The average time in milliseconds for the Gateway daemon to respond to API (ECI) and XA requests from remote clients including network I/O time. Successful and failed requests are included. This value is inclusive of the Gateway response time, as provided by the corresponding GD_IAVRESP statistic.	0	Integer
GD_IHAEXIT	The number of times the CICS request exit was called.	0	Integer
GD_ILUWTXNC	The number of extended LUW-based transactions that were committed. This statistic returns information about 1–phase commit transactions.	0	Integer
GD_ILUWTXNR	The number of extended LUW-based transactions that were rolled back. This statistic returns information about 1–phase commit transactions.	0	Integer
GD_IREQDATA	The amount of request data in bytes received from client applications. All requests are included.	0	Long
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_IXACOMP	The number of completed XA transactions that were started by another Gateway daemon in the group. This is an interval statistic. See also the description of the GD_LXACOMP lifetime statistic.	0	Integer
GD_IXAREQ	The number of XA requests processed for XA transactions.	0	Integer
GD_IXATXNC	The number of XA commit requests that were successfully processed.	0	Integer

Table 30. Gateway daemon statistics (continued)

ID	Description	Default value	Data type
GD_IXATXNHI	The peak number of XA transactions that have been in flight at the same time.	The default is the value for GD_CXATXN (current number of inflight XA transactions)	Integer
GD_IXATXNR	The number of XA rollback requests that were successfully processed.	0	Integer
GD_LALLREQ	The number of API calls (ECI, ESI) and XA requests that have been processed. Successful and failed requests are included. Administrative requests and handshakes are excluded.	0	Integer
GD_LAVRESP	The average time in milliseconds for the Gateway daemon to respond to API (ECI, ESI) and XA requests from remote clients. Successful and failed requests are included. This value is inclusive of the CICS response time, as provided by the corresponding CS_LAVRESP statistic.	0	Integer
GD_LAVRESPIO	The average time in milliseconds for the Gateway daemon to respond to API (ECI) and XA requests from remote clients including network I/O time. Successful and failed requests are included. This value is inclusive of the Gateway response time, as provided by the corresponding GD_LAVRESP statistic.	0	Integer
GD_LHAEXIT	The number of times the CICS request exit was called.	0	Integer
GD_LLUWTXNC	The number of extended LUW-based transactions that were committed. This statistic returns information about 1–phase commit transactions.	0	Integer
GD_LLUWTXNR	The number of extended LUW-based transactions that were rolled back. This statistic returns information about 1–phase commit transactions.	0	Integer
GD_LREQDATA	The amount of request data in bytes received from client applications. All requests are included.	0	Long
GD_LRESPDATA	The amount of response data in bytes sent to client applications. All responses are included.	0	Long
GD_LRUNTIME	The length of time in seconds since the Gateway daemon successfully initialized.	0	Long
GD_LSYNCFAIL	The number of SYNCONRETURN transactions that have failed for the duration of the Gateway daemon process.	0	Integer
GD_LSYNCTXN	The number of successful SYNCONRETURN transactions.	0	Integer

Table 30. Gateway daemon statistics (continued)

ID	Description	Default value	Data type
GD_LXACOMP	The number of completed XA transactions that were started by another Gateway daemon in the group. This is a lifetime statistic. See also the description of the GD_IXACOMP interval statistic.	0	Integer
GD_LXAREQ	The number of XA requests processed for XA transactions.	0	Integer
GD_LXATXNC	The number of XA commit requests that were successfully processed.	0	Integer
GD_LXATXNHI	The peak number of XA transactions that have been in flight at the same time.	0	Integer
GD_LXATXNR	The number of XA rollback requests that were successfully processed.	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_SNAME	The jobname of the Gateway daemon.	N/A	String
GD_SPLATFORM	The platform on which the CICS Transaction Gateway is running. Platform is one of the following: AIX, HP-UX (Itanium), Linux (Intel), Linux (POWER), Linux (zSeries), Solaris, Windows, z/OS, Unknown.	“Unknown”	String
GD_SDFLTSRV	The default CICS server for the CICS Transaction Gateway.	N/A	String
GD_SSTATEOD	The local time to be designated as the logical end of day by a Gateway daemon. At the logical end-of-day, all interval statistics are reset according to their defined default value. If the **statint** parameter has been set to an irregular value, the interval immediately prior to the **stateod** end-of-day is truncated. The value is in 24-hour HHMMSS format.	The value of the stateod parameter in the configuration file. If not specified in the configuration file the default value will be midnight, local time.	String

Table 30. 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

Logical CICS server (all) statistics:

The statistics listed here are members of the logical CICS server (all) resource group.

Table 31. Logical CICS server (all) statistics

ID	Description	Default value	Data type
LS_IALLREQ	The number of requests to logical CICS servers (successful and failed) that have been processed.	0	Integer
LS_ICOUNT	The number of logical CICS servers to which requests have been sent.	0	Integer
LS_ILIST	The list of logical CICS servers to which requests have been sent.	<empty string>	String
LS_LALLREQ	The number of requests to logical CICS servers (successful and failed) that have been processed.	0	Integer
LS_LCOUNT	The number of logical CICS servers to which requests have been sent.	0	Integer
LS_LLIST	The list of logical CICS servers to which requests have been sent.	<empty string>	String
LS_SCOUNT	The number of logical CICS servers defined in the configuration file.	0	Integer
LS_SLIST	The list of all logical CICS servers defined in the configuration file.	<empty string>	String

Logical CICS server (instance) statistics:

The statistics listed here are members of the logical CICS server (instance) resource group.

Table 32. CICS server (instance) statistics

ID	Description	Default value	Data type
LS*x*_IALLREQ	The number of requests to logical CICS server *x* (successful and failed) that have been processed.	0	Integer

Table 32. CICS server (instance) statistics (continued)

ID	Description	Default value	Data type
LS*x*_LALLREQ	The number of requests to logical CICS server *x* (successful and failed) that have been processed.	0	Integer
LS*x*_SLIST	The list of all CICS servers that logical CICS server *x* is mapped to in the configuration file.	N/A	String

System environment statistics:

The statistics listed here belong to the System Environment resource group.

Table 33. System Environment statistics

ID	Description	Default value	Data type
SE_CELOAL	The amount of currently allocated extended user private storage (in bytes) in the Gateway daemon address space. This value can increase or decrease.	0	Integer
SE_C31MAX	The real-time limit for the “used” storage indicated by SE_CELOAL. If SE_CELOAL approaches SE_C31MAX, then the Gateway daemon is at risk of failing with a Java `OutOfMemory` exception.	0	Integer
SE_CHEAPGCMIN	The Gateway daemon JVM heap size (in bytes) after the last garbage collection (GC).	0	Long
SE_IGCCOUNT	The number of garbage collection (GC) events.	0	Long
SE_IGCTIME	The length of time (in milliseconds) taken by the JVM for garbage collection (GC).	0	Long
SE_LGCCOUNT	The number of garbage collection (GC) events.	0	Long
SE_LGCTIME	The length of time (in milliseconds) taken by the JVM for garbage collection (GC).	0	Long
SE_SELIM	The amount of available extended user private storage (in bytes) in the Gateway daemon address space. This amount is less than or equal to the amount of storage specified by the job REGION parameter.	0	Integer
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 34. Protocol handler statistics

ID	Description	Default value	Data type
PH_SBINDSSL	The address or host name to which the SSL protocol handler is bound. This statistic does not contain a value if the protocol handler is not enabled.	none	String
PH_SBINDTCP	The address or host name to which the TCP protocol handler is bound. This statistic does not contain a value if the protocol handler is not enabled.	none	String
PH_SPORTSSL	The SSL protocol handler port number, or -1 if the protocol is not enabled.	-1	Integer
PH_SPORTTCP	The TCP protocol handler port number, or -1 if the protocol is not enabled.	-1	Integer

Worker thread statistics:

The statistics listed here belong to the worker thread resource group.

Table 35. Worker thread statistics

ID	Description	Default value	Data type
WT_CALLOC	The current number of worker threads that are being used by connection managers. Another way of viewing this value is the number of worker threads processing requests.	0	Integer
WT_CCURR	The current number of worker threads created.	0	Integer
WT_IALLOCHI	The peak number of worker threads concurrently allocated to connection manager threads. This number represents a high water mark for WT_CALLOC.	<WT_CALLOC>	Integer
WT_ITIMEOUTS	The number of times the Gateway daemon failed to allocate a worker thread to a connection manager within the defined **`workertimeout`** length of time.	0	Integer
WT_LTIMEOUTS	The number of times the Gateway daemon failed to allocate a worker thread to a connection manager within the defined **`workertimeout`** length of time.	0	Integer
WT_SINIT	The initial number of worker threads **`initworker`** created by the Gateway daemon.	0	Integer
WT_SMAX	The maximum number of parallel requests **`maxworker`** that the Gateway daemon can process.	0	Integer A value of -1 indicates no limit.

Using the statistics

This information classifies statistics into different categories, according to how they are most likely to be used. Some statistics are in more than one category.

Statistics for tuning and capacity planning

Look at these key statistics when analyzing the performance of the CICS Transaction Gateway.

Capture statistics when the CICS Transaction Gateway is operating under a number of different operating conditions. This will help you understand changes that might affect the performance of the system.

`CM_CALLOC`
The current number of connection manager threads allocated to clients.

`CM_CCURR`
The current number of connection manager threads created. If this value is greater than the configuration parameter **`initconnect`**, it signifies the peak number of remote clients connected at any one time. This value cannot exceed the maximum number of connection managers defined in CM_SMAX.

`CM_CWAITING`
The current number of connection managers waiting for a worker thread to become available. This statistic shows the number of requests that are queuing in the Gateway daemon. It is usually low or zero in a well-tuned Gateway daemon. If it is higher than expected, consider increasing the **`maxworker`** configuration parameter.

`CM_IALLOC`
The number of allocations for connection manager threads representing the number of connections that have been established from remote clients. A low value represents efficient connection reuse.

`CM_IALLOCHI`
The peak number of connection manager threads concurrently allocated to client applications. This number represents a high water mark for CM_CALLOC.

`CM_ICREATED`
The number of connection manager threads created.

`CM_ITIMEOUTS`
The number of times that the Gateway daemon failed to allocate a connection manager thread to a client application within the defined **`connecttimeout`** length of time.

`CM_LALLOC`
The number of allocations for connection manager threads representing the number of connections that have been established from remote clients. A stable value represents efficient connection reuse.

`CM_LTIMEOUTS`
The number of times that the Gateway daemon failed to allocate a connection manager thread to a client application within the defined (connecttimeout) length of time. This statistic shows the number of incoming connection requests that have been refused. It is usually low or zero in a well-tuned Gateway daemon. If it is high, consider increasing the **`connecttimeout`** or **`maxconnect`** configuration parameters.

`CM_SMAX`
The maximum number of connection manager threads **`maxconnect`** that can possibly be created and allocated by the Gateway daemon. This value limits the number of Java clients that can be connected at any one time.

CS_CALLOC
The current number of allocated EXCI pipes across all CICS servers. Compare this value to the number of pipes available on this CICS Transaction Gateway; see CS_SLOGONLIM. If the number of pipes is close to the maximum available, it indicates that your CICS Transaction Gateway system is running near full capacity. Check that your CICS system can run with the maximum available 250 EXCI available pipes. If the number of EXCI pipes cannot support your workload, you might need to make more EXCI pipes available by running with multiple Gateway daemons.

There are some other reasons why you might be running with many EXCI pipes:

- The CICS transactions are relatively long-running and therefore occupy EXCI connections for longer.
- Your CICS Transaction Gateway system is using the pipe reuse model CTG_PIPE_REUSE=ALL, and is communicating with more than one CICS server. This model gives optimum performance. However, after a pipe has been allocated, it remains allocated to this particular server and is not available for use with other CICS servers. Consider using pipe model CTG_PIPE_REUSE=ONE or direct all CICS Transaction Gateway requests into a single CICS TOR; this can then route these requests into other CICS servers.

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_IREALLOC
The number of times that a worker thread has to deallocate its existing pipe and reallocate a new pipe because the existing pipe points to a different CICS server from that required by the request. This statistic applies only when the environment variable CTG_PIPE_REUSE is set to ONE. Cases where the pipe for the worker thread was deallocated and reallocated because the CICS server was not available are excluded.

CS_LCOUNT
The number of CICS servers to which requests have been sent. This number equals the number of CICS servers in CS_LLIST. The value of CS_LCOUNT can be useful when trying to understand EXCI pipe shortage problems. See CS_CALLOC.

CS_LREALLOC
The number of times that a worker thread has to deallocate its existing pipe and reallocate a new pipe because the existing pipe points to a different CICS server from that required by the request. This statistic is useful when determining how many additional EXCI pipe deallocate and allocate calls have occurred when using the pipe model EXCI_PIPE_REUSE=ONE. The performance overhead of using EXCI_PIPE_REUSE=ONE is small in the context of typical overall transaction costs, but if this value is high relative to the total number of requests (CS_LALLREQ), consider using the pipe reuse model CTG_PIPE_REUSE=ALL.

CS_SLOGONLIM
The maximum EXCI pipe allocation for each MVS address space. This value is the equivalent to the CICS subsystem initialization parameter LOGONLIM. You can use this value to verify the maximum number of EXCI pipes that are available to your CICS Transaction Gateway.

CS*X*_CALLOC
The current number of allocated EXCI pipes for CICS server *x*. See CS_CALLOC.

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.

If the CTG_PIPE_REUSE environment variable is set to ONE, increasing **maxworker** to the maximum number of EXCI connections available can allow for a higher throughput.

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.

Related information:

"Environment variables: local and remote mode" on page 100
Environment variables available for use with local mode and remote mode topologies.

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.

CS_IALLOCFAIL
The number of times the Gateway daemon has tried and failed to allocate a pipe to a CICS server.

CS_IAVRESP
The average time taken (in milliseconds) for a connected CICS server to respond to the Gateway daemon over the course of this interval.

CS_ICOMMSFAIL
The number of times communication with a CICS server has failed after a connection has already been established, or when there has been a failure with the link during communication with the server.

CS_LALLOCFAIL
The number of times the Gateway daemon has tried and failed to allocate a pipe to a CICS server. If this value is high, consider increasing the LOGONLIM or modifying the pipe allocation model using the variable CTG_PIPE_REUSE=ONE. For more details, refer to the value for CSX_LALLOCFAIL.

CS_LCOMMSFAIL
The number of times communication with a CICS server has failed after a connection has already been established, or when there has been a failure with the link during communication with the server. In EXCI, a pipe was available but the request failed.

CS*X*_IALLOCFAIL
The number of times the Gateway daemon has tried and failed to allocate a pipe to CICS server *x*.

CS*X*_IAVRESP and CS*X*_LAVRESP
The average time taken (in milliseconds) for connected CICS server *x* to respond to the Gateway daemon.

CS*X*_LALLOCFAIL
The number of times the Gateway daemon has tried and failed to allocate a pipe to CICS server *x*.

GD_CHEALTH
The current health of communications between the Gateway daemon and CICS. For more details, see “Health reporting” on page 155.

GD_IAVRESP and GD_LAVRESP
The average time taken in milliseconds for the Gateway daemon to respond to API (ECI, ESI) 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.

CS*X*_CALLOC
The current number of allocated EXCI pipes for CICS server *x*. See CS_CALLOC for more information.

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.

If the CTG_PIPE_REUSE environment variable is set to ONE, increasing **maxworker** to the maximum number of EXCI connections available can allow for a higher throughput.

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.

CS_IALLREQ
The number of requests to CICS servers (successful and failed) that have been processed.

CS_IREQDATA
The amount of request data (in bytes) sent to connected CICS servers. This amount includes both application and CICS protocol data.

CS_IRESPDATA
The amount of response data (in bytes) received from connected CICS servers. This amount includes both application and CICS protocol data.

CS_LALLREQ
The number of requests to CICS servers (successful and failed) that have been processed.

CS_LREQDATA
The amount of request data (in bytes) sent to connected CICS servers. This amount includes both application and CICS protocol data. For IPIC, the data comprises COMMAREA and CICS headers.

CS_LRESPDATA
The amount of response data (in bytes) received from connected CICS servers. This amount includes both application and CICS protocol data. For IPIC, the data comprises COMMAREA and CICS headers.

CS*X*_IALLREQ
The number of requests to CICS server *x* (successful and failed) that have been processed.

CS*X*_IREQDATA
The amount of request data (in bytes) sent to connected CICS server *x*. This amount includes both application and CICS protocol data.

CS*X*_IRESPDATA
The amount of response data (in bytes) received from connected CICS server *x*. This amount includes both application and CICS protocol data.

CS*X*_LALLREQ
The number of requests to CICS server *x* (successful and failed) that have been processed.

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 IPIC, the data comprises COMMAREA and CICS headers.

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 IPIC, the data comprises COMMAREA and CICS headers.

GD_IALLREQ
The number of API calls (ECI, ESI) 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_IXATXNC
The number of XA commit requests that were successfully processed.

GD_IXATXNR
The number of XA rollback requests that were successfully processed.

GD_LALLREQ
The number of API calls (ECI, ESI) 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.

GD_LXATXNC
The number of XA commit requests that were successfully processed.

GD_LXATXNR
The number of XA rollback requests that were successfully processed.

Recording statistics to SMF

CICS Transaction Server, TXSeries, and WebSphere support the recording of statistics to SMF. The z/OS based products write SMF records. CICS Transaction Gateway provides similar function on the z/OS platform using type 111 SMF record format.

End-of-day and shutdown statistics

An attempt is always made to write end-of-day and shutdown statistics to SMF regardless of whether the **`statsrecording`** parameter is set to `on` or `off`. A message is logged if this attempt fails.

Interval correlation number

Interval statistics are captured at predefined intervals by issuing requests for them to be recorded at the correct time.

The interval statistics component periodically requests the SMF recording facility to record statistics information. The interval statistics component takes a snapshot of all current statistics and sends them to the SMF recorder. The SMF recorder formats the statistics into the documented SMF record format. This data is then sent to the SMF facility. Every time the interval statistics component issues a request to record statistics, a unique number is generated. This number is known as the interval correlation number and is unique, for a specific CICS Transaction Gateway instance, over the lifetime of that CICS Transaction Gateway.

The interval correlation number is included as an external field in every SMF record header.

The SMF subsystem allows a maximum of 30 KB per record. If the formatted data for a statistics request exceeds 30 KB, the data is split across multiple SMF records. Each SMF record is given the same interval correlation number. This correlation allows the records relevant to an individual statistics interval to be linked.

Correlating multiple SMF records within the same interval

The timestamp and datestamp fields, CTG_COLTIME and CTG_COLDATE, are used to correlate records for a specific CICS Transaction Gateway APPLID. A CICS Transaction Gateway instance cannot have intervals of less than 1 minute, so these fields can be used to identify the record. This correlation is valid if the CICS Transaction Gateway is stopped and restarted, whereas the interval count is reset at a stop and restart.

You do not need multiple records unless you have a large number of servers, because CICS Transaction Gateway writes data of greater than 30 KB in size.

For more information about configuring recording statistics to SMF, see “Configuring monitoring and statistics” on page 151

Correlating multiple SMF records from the same Gateway daemon

The CTG_STARTDATE and CTG_STARTTIME fields are used to correlate records for a specific instance of a CICS Transaction Gateway daemon. The fields are described in “SMF product section data structure” on page 341

Configuration for recording statistics to SMF

You can record statistics data from CICS Transaction Gateway at predefined intervals to the System Management Facility (SMF) on z/OS.

To set statistics recording to SMF manually, set the **`statsrecording`** parameter in the GATEWAY section of the Gateway daemon configuration file (ctg.ini) to `on`:

```
statsrecording=on
```

Valid values are `on` or `off`. Any other value results in an error sent to the JES log and CICS Transaction Gateway fails to start. Like the other Boolean parameters in the configuration file this parameter and the values it takes are not case-sensitive.

READ access to the BPX.SMF facility

To write to SMF, the user ID under which the CICS Gateway daemon runs, must be permitted READ access to the BPX.SMF facility. An example of the syntax is shown here:

```
PERMIT BPX.SMF CLASS(FACILITY) ACCESS(READ) ID(USERID)
```

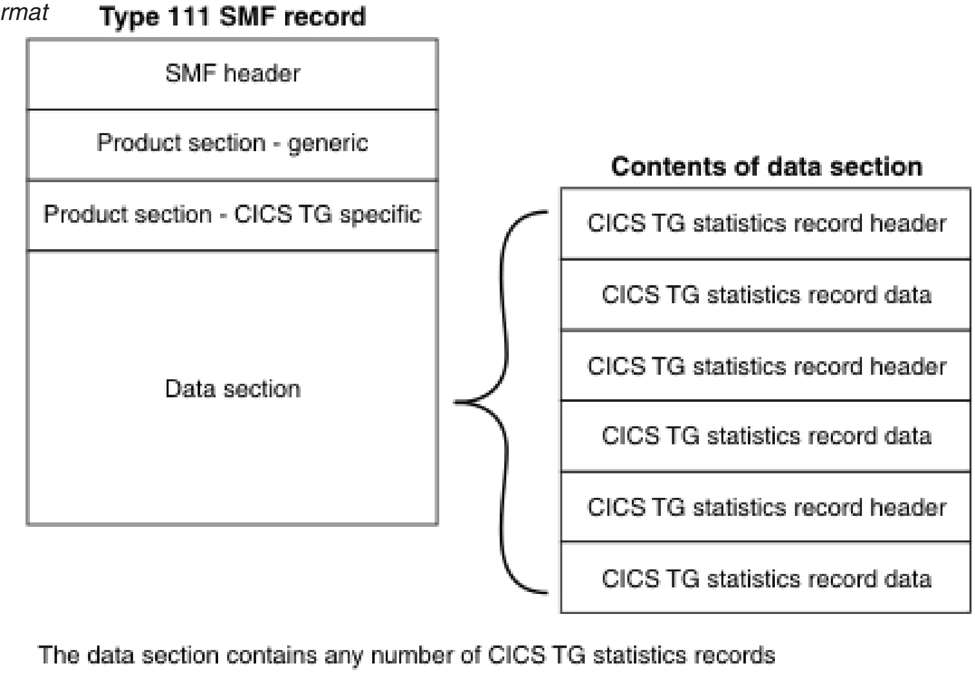
You must configure this permission during the installation and upgrade processes. For further details, see the RACF and UNIX System Services (USS) documentation.

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

SMF records

When you install CICS Transaction Gateway, a C header file and a z/OS assembler copy macro are provided to define the SMF record structure. The C header file is *CTGSMF*, located in the *SCTGINCL* data set and the z/OS assembler copy macro is file *CTGSMFA*, located in the SCTGMAC data set.

Figure 39. SMF record format


Type 111 SMF record
SMF header
Product section - generic
Product section - CICS TG specific
Data section
Contents of data section
CICS TG statistics record header
CICS TG statistics record data
CICS TG statistics record header
CICS TG statistics record data
CICS TG statistics record header
CICS TG statistics record data
The data section contains any number of CICS TG statistics records

SMF header data structure:

The format of the header for the SMF record for type 111 records is fixed and is common to SMF records from other CICS products. The Gateway daemon fills in the following fields that are marked with "*". The SMF subsystem fills in the rest of the fields. User applications are not required to fill in any SMF fields.

The entire data structure is initialized to binary zeros before setting any fields. All fields that are not filled have the default value of zero.

Table 36. SMF header data structure

Field name	Type	Offset	Length(bytes)	Description	Default value
SMF111_LEN*	Integer	0	2	Record length	32756 is the maximum
SMF111_SEG	Integer	2	2	Segment descriptor	NULL
SMF111_FLG*	Bit field	4	1	Operating system indicator. The Gateway daemon sets this value to 0xC0 to indicate that the CICS TG record supports triplets.	0xC0
SMF111_RTY*	Integer	5	1	Record type	0x6F (111 reserved for CICS TG)
SMF111_TIME	Integer	6	4	Time Record moved to SMF (100ths of a second)	Filled in by SMF
SMF111_SDTE	Packed	10	4	Date moved to SMF. PL4 Julian format. (OCYYDDD+)	Filled in by SMF
SMF111_SID	Char	14	4	System identification	Filled in by SMF
SMF111_SSI*	Char	18	4	Subsystem identification	CTGZ
SMF111_STY*	Integer	22	2	Record subtype. CICS Transaction Gateway uses only one subtype. This value is always zero.	0x0000
SMF111_TRN*	Integer	24	2	Number of triplets	2
		26	2	Reserved	
SMF111_APS*	Integer	28	4	Offset to product section	
SMF111_LPS*	Integer	32	2	Length of product section	Size of this structure
SMF111_NPS*	Integer	34	2	Number of product sections	1
SMF111_ASS*	Integer	36	4	Offset to DATA section	
SMF111_ASL*	Integer	40	2	Length of DATA section	The total length of all the CICS TG records included in the data section.
SMF111_ASN*	Integer	42	2	Number of DATA sections	1

SMF product section data structure:

The Gateway daemon fills in all the fields in the generic product section for the SMF record for type 111 records. Parts of the record are of fixed format and common to all SMF products. Some parts of this record are specific to CICS Transaction Gateway and have the prefix "CTG".

Table 37. SMF product section data structure

Field name	Type	Offset	Length (bytes)	Description	Default value
SMF111_VRM	Integer	0	2	Record version (0x0VRM) • Each letter represents a numeric digit from values 0 to 9 • V = Version • R = Release • M = Maintenance	Version 7.1.0 of CICS Transaction Gateway would be represented as 0x0710
SMF111_PRN	Char	2	8	Generic product name	CICSTGGW
SMF111_SPN	Char	10	8	Gateway daemon identifier	The CICS Transaction Gateway APPLID is used. If the APPLID is not set, the Jobname is used.
SMF111_MFL	Integer	18	2	Record maintenance indicator	0
	Reserved	20	2		NULL
CTG_STATTYPE	Integer	22	1	Statistics type. This event drives the statistic to be recorded	• 0x00 for interval • 0x01 for end-of-day • 0x02 for shutdown
	Reserved	23	1		NULL
CTG_COLDATE	Packed	24	4	Collection date (0CYYDDD+) Local time	Set to the time that the Gateway daemon requests that records for a statistics interval are cut. This value represents the time that the interval is initially requested. Time to process the interval data and perform the I/O is not included. The value corresponds to the **`statint`** and **`stateod`** values in the configuration file. The date that the record is written is filled in by SMF in SMF111_SMFSTDTE.
CTG_COLTIME	Packed	28	4	Collection time (00HHMMSS) Local time. The last byte is used; it is not reserved or set to +.	Set to the time that the Gateway daemonrequests that records for a statistics interval are cut. This value represents the time that the interval is initially requested. Time to process the interval data and perform the I/O is not included. The value corresponds to the **`statint`** and **`stateod`** values in the configuration file. The time that the record is written is filled in by SMF in SMF111_SMFTME.

Table 37. SMF product section data structure (continued)

Field name	Type	Offset	Length (bytes)	Description	Default value
CTG_LOCOFFSET	Signed Integer	32	4	Offset from GMT to local time in seconds. Signed integer.	Number to add to the system local time to derive GMT
CTG_LSTRESET	Integer	36	4	Last reset time or initialization time (in seconds)	The period in time in seconds since the last reset. If the first time stats have been issued in the lifetime of the CICS Transaction Gateway this value represents time since initialization. The reset is performed each time that interval statistics are cut.
CTG_INTERVAL	Integer	40	4	Interval seconds	Length of time remaining before the next interval
CTG_INTVCOUNT	Integer	44	4	Interval number	A sequence number that increments each time an interval is requested. The scope of this is within a specific Gateway daemoninstance, for the lifetime of that Gateway daemon. The first interval is defined as 1.
CTG_STARTDATE	Packed	48	4	Gateway start date (0CYYDDD+) Local time	Set to the date that the Gateway daemon started.
CTG_STARTTIME	Packed	52	4	Gateway start time (00HHMMSS) Local time. The last byte is used; it is not reserved or set to +.	Set to the time that the Gateway daemon started.

Related concepts:

"SMF date format - byte packed date format" on page 353
Definition of the format used to represent dates (OCYYDDD+) for SMF.

"SMF data format - sample values" on page 354
Sample values for CTG_COLTIME and CTG_LOCOFFSET.

Statistics record header:

The data section contains a number of statistics records. Each statistics record has a header. The header is common to all CICS Transaction Gateway statistics records. You can work out the type of data that follows the record from the `CTG_RECORDID` and `CTG_RECVER` fields.

Each CICS Transaction Gateway statistics record has one instance except for CSX statistics for EXCI and IPIC, for which multiple instances of the records occur in one SMF record, because each server has one CSX statistics record, uniquely identified by the netname of the server. The netname (CTG_CSX_SAPPLID) is unique for each CSX statistics record.

Table 38. CICS Transaction Gateway Statistics record header

Field name	Type	Offset	Length (bytes)	Description	Default value
CTG_DATALEN	Integer	0	2	Length of record	Length of this header record + the length of the data that follows
CTG_RECORDID	Integer	2	2	Record ID resource group	Each CICS TG statistics resource group has a numeric ID allocated to it: • CM 0x00 • CS 0x01 • CSx for EXCI 0x02 • GD 0x03 • PH 0x04 • WT 0x05 • SE 0x06 • CSx for IPIC 0x07
CTG_RECVER	Integer	4	1	Data section version	A combination of CTGRECID and CTGRECVER controls the version of the structure. Because structures are fixed length, you change CTGRECVER every time a statistic is added to a resource group (in between releases).
	Reserved	5	3	NOT DISPLAYED	Pads length up to a multiple of 8

Connection Manager (CM) statistics SMF data:

CICS Transaction Gateway CM statistics use this format for writing SMF records.

For a full description of these statistics see “Connection manager statistics” on page 320.

Table 39. Connection Manager (CM) statistics data

Field name	Statistic ID	Description	Type	Offset	Length (bytes)
CTG_CM_CALLOC	CM_CALLOC	Currently allocated connection managers	Integer	0	4
CTG_CM_CCURR	CM_CCURR	Current number of connection managers	Integer	4	4
CTG_CM_CWAITING	CM_CWAITING	Number of connection managers waiting	Integer	8	4
CTG_CM_LTIMEOUTS	CM_LTIMEOUTS	Number of times connect timeout reached	Integer	12	4
CTG_CM_SINIT	CM_SINIT	Initial number of connection managers	Integer	16	4

Table 39. Connection Manager (CM) statistics data (continued)

Field name	Statistic ID	Description	Type	Offset	Length (bytes)
CTG_CM_SMAX	CM_SMAX	Maximum number of connection managers	Signed Integer	20	4
CTG_CM_ITIMEOUTS	CM_ITIMEOUTS	Number of times connect timeout reached	Integer	24	4
CTG_CM_IALLOCHI	CM_IALLOCHI	Peak number of allocated connection manager threads	Integer	28	4
CTG_CM_ICREATED	CM_ICREATED	Number of connection manager threads created	Integer	32	4
CTG_CM_IALLOC	CM_IALLOC	Number of times a connection manager thread was allocated	Integer	36	4
CTG_CM_LALLOC	CM_LALLOC	Number of allocations for connection manager threads representing the number of connections that have been established from remote clients.	Integer	40	4
			Reserved	44	4

CICS Server all (CS) statistics SMF data:

CICS Transaction Gateway CS statistics use this format for writing SMF records.

For a full description of these statistics see "CICS server (all) statistics" on page 321.

Table 40. CICS Server all (CS) statistics data

Field name	Statistic ID	Description	Type	Offset	Length (bytes)
CTG_CS_CALLOC	CS_CALLOC	Number of EXCI pipes allocated	Integer	0	4
CTG_CS_LALLOCFAIL	CS_LALLOCFAIL	Number of EXCI pipe allocate failures	Integer	4	4
CTG_CS_LALLREQ	CS_LALLREQ	Number of requests processed	Integer	8	4
CTG_CS_LCOMMSFAIL	CS_LCOMMSFAIL	Number of CICS communication failures	Integer	12	4
CTG_CS_LCOUNT	CS_LCOUNT	Number of CICS servers	Integer	16	4
CTG_CS_LREALLOC	CS_LREALLOC	Number of EXCI pipe reallocations	Integer	20	4
CTG_CS_SLOGONLIM	CS_SLOGONLIM	EXCI pipe limit	Integer	24	4
CTG_CS_SNETNAME	CS_SNETNAME	EXCI NETNAME	Char	28	8

Table 40. CICS Server all (CS) statistics data (continued)

Field name	Statistic ID	Description	Type	Offset	Length (bytes)
CTG_CS_IALLOCFAIL	CS_IALLOCFAIL	Number of EXCI pipe allocate failures	Integer	36	4
CTG_CS_IALLREQ	CS_IALLREQ	Number of requests processed	Integer	40	4
CTG_CS_ICOMMSFAIL	CS_ICOMMSFAIL	Number of CICS communication failures	Integer	44	4
CTG_CS_ICOUNT	CS_ICOUNT	Number of CICS servers	Integer	48	4
CTG_CS_IREALLOC	CS_IREALLOC	Number of EXCI pipe reallocations	Integer	52	4
CTG_CS_IREQDATA	CS_IREQDATA	Amount of CICS request data	Integer	56	8
CTG_CS_LREQDATA	CS_LREQDATA	Amount of CICS request data	Integer	64	8
CTG_CS_IRESPDATA	CS_IRESPDATA	Amount of CICS response data	Integer	72	8
CTG_CS_LRESPDATA	CS_LRESPDATA	Amount of CICS response data	Integer	80	8
CTG_CS_SCOUNT	CS_SCOUNT	Number of defined CICS servers	Integer	88	4
CTG_CS_ICONNFAIL	CS_ICONNFAIL	Number of connect failures	Integer	92	4
CTG_CS_LCONNFAIL	CS_LCONNFAIL	Number of connect failures	Integer	96	4
CTG_CS_ILOSTCON	CS_ILOSTCON	Number of lost connections	Integer	100	4
CTG_CS_LLOSTCON	CS_LLOSTCON	Number of lost connections	Integer	104	4
CTG_CS_LIDLETIMEOUT	CS_LIDLETIMEOUT	Number of timed out connections	Integer	108	4
CTG_CS_CSESSCURR	CS_CSESSCURR	Number of IPIC sessions in use	Integer	112	4
CTG_CS_CSESSMAX	CS_CSESSMAX	Number of negotiated IPIC sessions	Integer	116	4
CTG_CS_LSESSFAIL	CS_LSESSFAIL	Number of IPIC session failures	Integer	120	4
CTG_CS_ISESSFAIL	CS_ISESSFAIL	Number of IPIC session failures	Integer	124	4
CTG_CS_CWAITING	CS_CWAITING	Number of requests waiting on a response	Integer	128	4
CTG_CS_IIDLETIMEOUT	CS_IIDLETIMEOUT	Number of timed out connections	Integer	132	4
CTG_CS_IAVRESP	CS_IAVRESP	Average CICS response time	Integer	136	4
CTG_CS_LAVRESP	CS_LAVRESP	Average CICS response time	Integer	140	4

CICS Server Instance (CS*x*) SMF statistics for EXCI:

Each CICS Transaction Gateway statistics record has one instance, except for CS*x* statistics for EXCI and IPIC, for which multiple instances of the records occur in one SMF record, because each server has one CS*x* statistics record uniquely identified by the netname of the server. The netname (CTG_CS*x*_SAPPLID) is unique for each CS*x*statistics record.If the server name contains one or more underscore characters (_), they are replaced by minus characters (-).

One record is written in an SMF interval, when the data collected is less than 30 KB. If the data exceeds 30 KB, multiple records are written. Correlation of multiple SMF records is achieved using the COLTIME field, in the product section data structure, because this is unique for each interval.

For a full description of these statistics see "CICS server (instance) statistics" on page 323.

Table 41. CICS Server Instance (CSx) statistics for EXCI

Field name	Statistic ID	Description	Type	Offset	Length (bytes)
CTG_CS*x*_SAPPLID	Not applicable	Name of CICS server	Char	0	8
CTG_CS*x*_CALLOC	CS*x*_CALLOC	Number of EXCI pipes allocated	Integer	8	4
CTG_CS*x*_LALLOCFAIL	CS*x*_LALLOCFAIL	Number of EXCI pipe allocate failures	Integer	12	4
CTG_CS*x*_LALLREQ	CS*x*_LALLREQ	Number of requests processed	Integer	16	4
CTG_CS*x*_IALLOCFAIL	CS*x*_IALLOCFAIL	Number of EXCI pipe allocate failures	Integer	20	4
CTG_CS*x*_IALLREQ	CS*x*_IALLREQ	Number of requests processed	Integer	24	4
CTG_CS*x*_IAVRESP	CS*x*_IAVRESP	Average CICS response time	Integer	28	4
CTG_CS*x*_LAVRESP	CS*x*_LAVRESP	Average CICS response time	Integer	32	4
CTG_CS*x*_LCOMMSFAIL	CS*x*_LCOMMSFAIL	Number of CICS communication failures	Integer	36	4
CTG_CS*x*_IREQDATA	CS*x*_IREQDATA	Amount of CICS request data	Integer	40	8
CTG_CS*x*_LREQDATA	CS*x*_LREQDATA	Amount of CICS request data	Integer	48	8
CTG_CS*x*_IRESPDATA	CS*x*_IRESPDATA	Amount of CICS response data	Integer	56	8
CTG_CS*x*_LRESPDATA	CS*x*_LRESPDATA	Amount of CICS response data	Integer	64	8
CTG_CS*x*_SPROTOCOL	CS*x*_SPROTOCOL	CICS server protocol	String	72	8
CTG_CS*x*_ICOMMSFAIL	CS*x*_ICOMMSFAIL	Number of CICS communication failures	Integer	80	4
CTG_CS*x*_CWAITING	CS*x*_CWAITING	Number of requests waiting on a response	Integer	84	4

CICS Server Instance (CS*x*) SMF statistics for IPIC:

Each CICS Transaction Gateway statistics record has one instance, except for CS*x* statistics for EXCI and IPIC, for which multiple instances of the records occur in one SMF record, because each server has one CS*x* statistics record uniquely identified by the netname of the server. The server name that is configured in the configuration file (CTG_CS*x*_SININAME) is unique for each CS*x* statistics record. If the server name contains one or more underscore characters (_), they are replaced by minus characters (-).

One record is written in an SMF interval, when the data collected is less than 30 KB. If the data exceeds 30 KB, multiple records are written. Correlation of multiple SMF records is achieved using the COLTIME field, in the product section data structure, because this is unique for each interval.

For a full description of these statistics see "CICS server (instance) statistics" on page 323.

Table 42. CICS Server Instance (CSx) statistics for IPIC

Field name	Statistic ID	Description	Type	Offset	Length (bytes)
CTG_CS*x*_SININAME	Not applicable	CICS server name	String	0	8
CTG_CS*x*_IREQDATA	CS*x*_IREQDATA	Amount of CICS request data	Integer	8	8
CTG_CS*x*_LREQDATA	CS*x*_LREQDATA	Amount of CICS request data	Integer	16	8
CTG_CS*x*_IRESPDATA	CS*x*_IRESPDATA	Amount of CICS response data	Integer	24	8
CTG_CS*x*_LRESPDATA	CS*x*_LRESPDATA	Amount of CICS response data	Integer	32	8
CTG_CS*x*_IALLREQ	CS*x*_IALLREQ	Number of requests processed	Integer	40	4
CTG_CS*x*_LALLREQ	CS*x*_LALLREQ	Number of requests processed	Integer	44	4
CTG_CS*x*_ICONNFAIL	CS*x*_ICONNFAIL	Number of connection failures	Integer	48	4
CTG_CS*x*_LCONNFAIL	CS*x*_LCONNFAIL	Number of lost connections	Integer	52	4
CTG_CS*x*_ILOSTCONN	CS*x*_ILOSTCONN	Number of lost connections	Integer	56	4
CTG_CS*x*_LLOSTCONN	CS*x*_LLOSTCONN	Number of lost connections	Integer	60	4
CTG_CS*x*_IIDLETIMEOUT	CS*x*_IIDLETIMEOUT	Number of timed out connections	Integer	64	4
CTG_CS*x*_LIDLETIMEOUT	CS*x*_LIDLETIMEOUT	Number of timed out connections	Integer	68	4
CTG_CS*x*_SIPADDR	CS*x*_SIPADDR	CICS server TCP/IP address	String	72	104
CTG_CS*x*_SIPPORT	CS*x*_SIPPORT	CICS server TCP/IP port	Integer	176	4

Table 42. CICS Server Instance (CSx) statistics for IPIC (continued)

Field name	Statistic ID	Description	Type	Offset	Length (bytes)
CTG_CS*x*_CSESSCURR	CS*x*_CSESSCURR	Number of IPIC sessions in use	Integer	180	4
CTG_CS*x*_ CSESSMAX	CS*x*_ CSESSMAX	Number of negotiated IPIC sessions	Integer	184	4
CTG_CS*x*_ SSESSMAX	CS*x*_ SSESSMAX	Number of requested IPIC sessions	Integer	188	4
CTG_CS*x*_LSESSFAIL	CS*x*_LSESSFAIL	Number of IPIC session failures	Integer	192	4
CTG_CS*x*_ISESSFAIL	CS*x*_ISESSFAIL	Number of IPIC session failures	Integer	196	4
CTG_CS*x*_SPROTOCOL	CS*x*_SPROTOCOL	CICS server protocol	String	200	8
CTG_CS*x*_LCOMMSFAIL	CS*x*_LCOMMSFAIL	Number of CICS communication failures	Integer	208	4
CTG_CS*x*_ICOMMSFAIL	CS*x*_ICOMMSFAIL	Number of CICS communication failures	Integer	212	4
CTG_CS*x*_CWAITING	CS*x*_CWAITING	Number of requests waiting on a response	Integer	216	4
CTG_CS*x*_IAVRESP	CS*x*_IAVRESP	Average CICS response time	Integer	220	4
CTG_CS*x*_LAVRESP	CS*x*_LAVRESP	Average CICS response time	Integer	224	4
CTG_CS*x*_CAPPLIDQ	CS*x*_CAPPLIDQ	APPLID qualifier of the connected CICS server	String	228	8
CTG_CS*x*_CAPPLID	CS*x*_CAPPLID	APPLID of the connected CICS server	String	236	8
			Reserved	244	4

Gateway daemon (GD) statistics SMF data:

CICS Transaction Gateway daemon statistics use this format for writing SMF records.

For a full description of these statistics see “Gateway daemon statistics” on page 325.

Table 43. Gateway daemon (GD) statistics data

Field name	Statistic ID	Description	Type	Offset	Length (bytes)
CTG_GD_CHEALTH	GD_CHEALTH	Gateway daemon health	Integer	0	4
CTG_GD_CSTATUS	GD_CSTATUS	Gateway daemon status	Char	4	16
CTG_GD_LALLREQ	GD_LALLREQ	Requests processed	Integer	20	4
CTG_GD_LLUWTXNC	GD_LLUWTXNC	Extended LUW transactions committed	Integer	24	4

Table 43. Gateway daemon (GD) statistics data (continued)

Field name	Statistic ID	Description	Type	Offset	Length (bytes)
CTG_GD_LLUWTXNR	GD_LLUWTXNR	Extended LUW transactions rolled back	Integer	28	4
CTG_GD_LRUNTIME	GD_LRUNTIME	Gateway daemon running time	Integer	32	8
CTG_GD_LSYNCTXN	GD_LSYNCTXN	Successful SYNCONRETURN transactions	Integer	40	4
CTG_GD_LXATXNC	GD_LXATXNC	XA commit requests successfully processed	Integer	44	4
CTG_GD_LXATXNR	GD_LXATXNR	XA rollback requests successfully processed	Integer	48	4
CTG_GD_SNAME	GD_SNAME	Gateway daemon name	Char	52	8
CTG_GD_IALLREQ	GD_IALLREQ	Requests processed	Integer	60	4
CTG_GD_IRUNTIME	GD_IRUNTIME	Interval running time	Integer	64	4
CTG_GD_ISYNCTXN	GD_ISYNCTXN	Successful SYNCONRETURN transactions	Integer	68	4
CTG_GD_IXATXNC	GD_IXATXNC	XA commit requests successfully processed	Integer	72	4
CTG_GD_IXATXNR	GD_IXATXNR	XA rollback requests successfully processed	Integer	76	4
CTG_GD_ILUWTXNC	GD_ILUWTXNC	Extended LUW transactions committed	Integer	80	4
CTG_GD_ILUWTXNR	GD_ILUWTXNR	Extended LUW transactions rolled back	Integer	84	4
CTG_GD_IAVRESP	GD_IAVRESP	Average Gateway daemon response time	Integer	88	4
CTG_GD_LAVRESP	GD_LAVRESP	Average Gateway daemon response time	Integer	92	4
CTG_GD_IREQDATA	GD_IREQDATA	Amount of client request data	Integer	96	8
CTG_GD_LREQDATA	GD_LREQDATA	Amount of client request data	Integer	104	8
CTG_GD_IRESPDATA	GD_IRESPDATA	Amount of client response data	Integer	112	8
CTG_GD_LRESPDATA	GD_LRESPDATA	Amount of client response data	Integer	120	8

Table 43. Gateway daemon (GD) statistics data (continued)

Field name	**Statistic ID**	**Description**	**Type**	**Offset**	**Length (bytes)**
CTG_GD_CSYNCTXN	GD_CSYNCTXN	SYNCONRETURN transactions	Integer	128	4
CTG_GD_CLUWTXN	GD_CLUWTXN	Extended LUW transactions	Integer	132	4
CTG_GD_CXATXN	GD_CXATXN	XA transactions	Integer	136	4
CTG_GD_LXAREQ	GD_LXAREQ	XA requests	Integer	140	4
CTG_GD_IXAREQ	GD_IXAREQ	XA requests	Integer	144	4
CTG_GD_SAPPLID	GD_SAPPLID	CICS TG APPLID	String	148	8
CTG_GD_SAPPLIDQ	GD_SAPPLIDQ	CICS TG APPLID qualifier	String	156	8
CTG_GD_SSTATINT	GD_SSTATINT	Length of the statistics interval HHMMSS	String	164	6
CTG_GD_SSTATEOD	GD_SSTATEOD	Logical end-of-day time HHMMSS	String	170	6
CTG_GD_CNEXTRESET	GD_CNEXTRESET	End of interval time HHMMSS	String	176	6
CTG_GD_LHAEXIT	GD_LHAEXIT	Times the CICS request exit was called	Integer	184	4
CTG_GD_IHAEXIT	GD_IHAEXIT	Times the CICS request exit was called	Integer	188	4
CTG_GD_LSYNCFAIL	GD_LSYNCFAIL	SYNCONRETURN transactions that have failed for the duration of the Gateway daemon process	Integer	192	4
CTG_GD_ISYNCFAIL	GD_ISYNCFAIL	SYNCONRETURN transactions that have failed in the current interval	Integer	196	4
CTG_GD_SDFLTSRV	GD_SDFLTSRV	The default CICS server for the CICS Transaction Gateway	String	200	8
CTG_GD_SHOSTNAME	GD_SHOSTNAME	The host name of the CICS Transaction Gateway computer	String	208	128
CTG_GD_LXACOMP	GD_LXACOMP	XA transactions completed for HA group	Integer	336	4
CTG_GD_IXACOMP	GD_IXACOMP	XA transactions completed for HA group	Integer	340	4
CTG_GD_LXATXNHI	GD_LXATXNHI	Peak number of in flight XA transactions	Integer	344	4

Table 43. Gateway daemon (GD) statistics data (continued)

Field name	**Statistic ID**	**Description**	**Type**	**Offset**	**Length (bytes)**
CTG_GD_IXATXNHI	GD_IXATXNHI	Peak number of in flight XA transactions	Integer	348	4
CTG_GD_LAVRESPIO	GD_LAVRESPIO	Average Gateway daemon response time with I/O	Integer	352	4
CTG_GD_IAVRESPIO	GD_IAVRESPIO	Average Gateway daemon response time with I/O	Integer	356	4

Protocol handler (PH) statistics SMF data:

CICS Transaction Gateway PH statistics have this format.

For a full description of these statistics see "Protocol handler statistics" on page 330.

Table 44. Protocol handler (PH) statistics data

Field name	**Statistic ID**	**Description**	**Type**	**Offset**	**Length (bytes)**
CTG_PH_SPORTSSL	PH_SPORTSSL	SSL protocol handler port number	Signed integer	0	4
CTG_PH_SPORTTCP	PH_SPORTTCP	TCP protocol handler port number	Signed integer	4	4

System environment (SE) statistics SMF data:

CICS Transaction Gateway SE statistics have this format.

For full descriptions of these statistics see "System environment statistics" on page 330,

Table 45. System environment (SE) statistics data

Field name	**Statistic ID**	**Description**	**Type**	**Offset**	**Length (bytes)**
CTG_SE_SELIM	SE_SELIM	Amount of available memory ELIM	Integer	0	4
CTG_SE_CELOAL	SE_CELOAL	Amount of used memory ELOAL	Integer	4	4
CTG_SE_CHEAPGCMIN	SE_CHEAPGCMIN	JVM heap size after GC	Integer	8	8
CTG_SE_SHEAPINIT	SE_SHEAPINIT	JVM initial heap size	Integer	16	8
CTG_SE_ SHEAPMAX	SE_ SHEAPMAX	JVM maximum heap size	Integer	24	8
CTG_SE_ IGCTIME	SE_ IGCTIME	JVM GC time	Integer	32	8
CTG_SE_ LGCTIME	SE_ LGCTIME	JVM GC time	Integer	40	8
CTG_SE_ IGCCOUNT	SE_ IGCCOUNT	JVM GC count	Integer	48	8
CTG_SE_ LGCCOUNT	SE_ LGCCOUNT	JVM GC count	Integer	56	8

Table 45. System environment (SE) statistics data (continued)

Field name	Statistic ID	Description	Type	Offset	Length (bytes)
CTG_SE_C31MAX	SE_C31MAX	Limit of used memory ELOAL	Integer	64	4

Worker thread (WT) statistics SMF data:

CICS Transaction Gateway WT statistics have this format.

For a full description of these statistics see “Worker thread statistics” on page 331.

Table 46. Worker thread (WT) statistics data

Field name	Statistic ID	Description	Type	Offset	Length (bytes)
CTG_WT_CALLOC	WT_CALLOC	Currently allocated worker threads	Char	0	4
CTG_WT_CCURR	WT_CCURR	Current number of worker threads	Integer	4	4
CTG_WT_LTIMEOUTS	WT_LTIMEOUTS	Number of times worker timeout reached	Integer	8	4
CTG_WT_SINIT	WT_SINIT	Initial number of worker threads	Integer	12	4
CTG_WT_SMAX	WT_SMAX	Maximum number of worker threads	Signed integer	16	4
CTG_WT_ITIMEOUTS	WT_ITIMEOUTS	Number of times worker timeout reached	Integer	20	4
CTG_WT_IALLOCHI	WT_IALLOCHI	Peak number of allocated worker threads	Integer	24	4
			Reserved	28	4

SMF date format - byte packed date format:

Definition of the format used to represent dates (OCYYDDD+) for SMF.

The numbers are written in +ve byte packed decimal.

'0C' is the number of centuries since 1900. All dates in the 21st century have '01'.

'YY' is the year.

'DDD' is the day of the year. The first day of the year is '1' and the last day of the year '366'.

'+' is hard coded as 0xC.

Examples

10 January 2007 = 0x0107010C

1 January 2001 = 0x0101001C

31December 2012 = 0x0112366C

SMF data format - sample values:

Sample values for CTG_COLTIME and CTG_LOCOFFSET.

Table 47. CTG_COLTIME

Interval cut at:	Value
00:00.00 local time	0x00000000
00:00.01 local time	0x00000001
23:59.59 local time	0x00235959
02:10.15 local time	0x00021015

The numbers are written in +ve byte packed decimal.

Table 48. CTG_LOCOFFSET

System time zone	Value
System time zone is GMT	CTG_LOCOFFSET= 0
System time zone is GMT +1	CTG_LOCOFFSET= 3600
System time zone is GMT -1	CTG_LOCOFFSET= -3600

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.

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.

Glossary

This glossary defines the terms and abbreviations used in CICS Transaction Gateway and in the information centers.

A

abnormal end of task (abend)
The termination of a task, job, or subsystem because of an error condition that recovery facilities cannot resolve.

Advanced program-to-program communication (APPC)
An implementation of the SNA/SDLC LU 6.2 protocol that allows interconnected systems to communicate and share the processing of programs. The Client daemon uses APPC to communicate with CICS systems.

APAR See *Authorized program analysis report*.

API See *application programming interface*.

APPC See *Advanced program-to-program communication*.

application programming interface (API)
A functional interface that allows an application program that is written in a high-level language to use specific data or functions of the operating system or another program.

APPLID

1. On CICS Transaction Gateway: The application identifier that is used to identify connections on the CICS server and tasks in a CICSplex. See also *APPLID qualifier* and *fully-qualified APPLID*.
2. On CICS Transaction Server: The name by which a CICS system is known in a network of interconnected CICS systems. CICS Transaction Gateway application identifiers do not need to be defined in SYS1.VTAMLST. The CICS APPLID is specified in the APPLID system initialization parameter.

APPLID qualifier
Optionally used as a high-level qualifier for the APPLID to form a fully-qualified APPLID. See also *APPLID* and *fully-qualified APPLID*.

ARM See *automatic restart manager*.

Authorized program analysis report (APAR)
A request for correction of a defect in a current release of an IBM-supplied program.

ATI See *automatic transaction initiation*.

attach In SNA, the request unit that flows on a session to initiate a conversation.

Attach Manager
The component of APPC that matches attaches received from remote computers to accepts issued by local programs.

autoinstall
A method of creating and installing resources dynamically as terminals log on, and deleting them at logoff.

automatic restart manager (ARM)
A z/OS recovery function that can improve the availability of specific batch jobs or started tasks, and therefore result in faster resumption of productive work.

automatic transaction initiation (ATI)
The initiation of a CICS transaction by an internally generated request, for example, the issue of an EXEC CICS START command or the reaching of a transient data trigger level. CICS resource definition can associate a trigger level and a transaction with a transient data destination. When the number of records written to the destination reaches the trigger level, the specified transaction is automatically initiated.

B

bean A definition or instance of a JavaBeans component. See also *JavaBeans*.

bean-managed transaction
A transaction where the JEE bean itself is responsible for administering transaction tasks such as committal or rollback. See also *container-managed transaction*.

BIND command
In SNA, a request to activate a session between two logical units (LUs).

business logic
The part of a distributed application that is concerned with the application logic rather than the user interface of the application. Compare with *presentation logic*.

C

CA See *certificate authority*.

CCIN The CCIN transaction is invoked by the Client daemon, for each TCP/IP or SNA connection established. CCIN installs a Client connection on the CICS server.

CCSID
Coded Character Set Identifier. A 16-bit number that includes a specific set of encoding scheme identifiers, character set identifiers, code page identifiers, and other information that uniquely identifies the coded graphic-character representation.

CTIN The CTIN transaction is invoked by the Client daemon to install a Client terminal definition on the CICS server.

callback
A way for one thread to notify another application thread that an event has happened.

certificate authority (CA)
In computer security, an organization that issues certificates. The certificate authority authenticates the certificate owner's identity and the services that the owner is authorized to use. It issues new certificates and revokes certificates from users who are no longer authorized to use them.

change-number-of-sessions (CNOS)
An internal transaction program that regulates the number of parallel sessions between the partner LUs with specific characteristics.

channel
A channel is a set of containers, grouped together to pass data to CICS. There is no limit to the number of containers that can be added to a channel, and the size of individual containers is limited only by the amount of storage that you have available.

CICS connectivity components
A generic reference to the Client daemon, EXCI, and the IPIC protocol.

CICS connectivity components
The Client daemon, the EXCI (External CICS Interface), and the IPIC (IP Interconnectivity) protocol are collectively called the 'CICS connectivity components'. The Client daemon handles the TCP/IP and the SNA protocols.

CICS Request Exit
An exit that is invoked by the CICS Transaction Gateway for z/OS at run time to determine which CICS server to use.

CICS server name
A defined server known to CICS Transaction Gateway.

CICS TS
Abbreviation of CICS Transaction Server.

class In object-oriented programming, a model or template that can be instantiated to create objects with a common definition and therefore, common properties, operations, and behavior. An object is an instance of a class.

CLASSPATH
In the execution environment, an environment variable keyword that specifies the directories in which to look for class and resource files.

Client API
The Client API is the interface used by Client applications to interact with CICS using the Client daemon. See External Call Interface, External Presentation Interface, and External Security Interface.

Client application
The client application is a user application written in a supported programming language that uses one or more of the CICS Transaction Gateways APIs.

Client daemon
The Client daemon manages TCP/IP and SNA connections to CICS servers on UNIX, Linux, and Windows. It processes ECI, EPI, and ESI requests, sending and receiving the appropriate flows to and from the CICS server to satisfy Client application requests. It can support concurrent requests to one or more CICS servers. The CICS Transaction Gateway initialization file defines the operation of the Client daemon and the servers and protocols used for communication.

client/server
Pertaining to the model of interaction in distributed data processing in which a program on one computer sends a request to a program on another computer and awaits a response. The requesting program is called a client; the answering program is called a server.

CNOS See *Change-Number-of-Sessions*.

code page
An assignment of hexadecimal identifiers (code points) to graphic characters. Within a given code page, a code point can have only one meaning.

color mapping file
A file that is used to customize the 3270 screen color attributes on client workstations.

COMMAREA
See *communication area*.

commit phase
The second phase in a XA process. If all participants acknowledge that they are prepared to commit , the transaction manager issues the commit request. If any participant is not prepared to commit the transaction manager issues a back-out request to all participants.

communication area (COMMAREA)
A communication area that is used for passing data both between programs within a transaction and between transactions.

Configuration file
A file that specifies the characteristics of a program, system device, server or network.

connection
In data communication, an association established between functional units for conveying information.

In Open Systems Interconnection architecture, an association established by a given layer between two or more entities of the next higher layer for the purpose of data transfer.

In TCP/IP, the path between two protocol application that provides reliable data stream delivery service.

In Internet, a connection extends from a TCP application on one system to a TCP application on another system.

container
A container is a named block of data designed for passing information between programs. A container is a "named COMMAREA" that is not limited to 32KB. Containers are grouped together in sets called channels.

container-managed transaction
A transaction where the EJB container is responsible for administration of tasks such as committal or rollback. See also *bean-managed transaction*.

control table
In CICS, a storage area used to describe or define the configuration or operation of the system.

conversation
A connection between two programs over a session that allows them to communicate with each other while processing a transaction.

conversation security
In APPC, a process that allows validation of a user ID or group ID and password before establishing a connection.

D

daemon
A program that runs unattended to perform continuous or periodic systemwide functions, such as network control. A daemon can be launched automatically, such as when the operating system is started, or manually.

data link control (DLC)
A set of rules used by nodes on a data link (such as an SDLC link or a token ring) to accomplish an orderly exchange of information.

DBCS See *double-byte character set*.

default CICS server
The CICS server that is used if a server name is not specified on an ECI, EPI, or ESI request. The default CICS server name is defined as a product wide setting in the configuration file (ctg.ini).

dependent logical unit
A logical unit that requires assistance from a system services control point (SSCP) to instantiate an LU-to-LU session.

deprecated
Pertaining to an entity, such as a programming element or feature, that is supported but no longer recommended, and that might become obsolete.

digital certificate
An electronic document used to identify an individual, server, company, or some other entity, and to associate a public key with the entity. A digital certificate is issued by a certificate authority and is digitally signed by that authority.

digital signature
Information that is encrypted with an entity's private key and is appended to a message to assure the recipient of the authenticity and integrity of the message. The digital signature proves that the message was signed by the entity that owns, or has access to, the private key or shared secret symmetric key.

distinguished name
The name that uniquely identifies an entry in a directory. A distinguished name is made up of attribute:value pairs, separated by commas. The format of a distinguished name is defined by RFC4514. For more information, see `http://www.ietf.org/rfc/rfc4514.txt`. See also *realm name* and *identity propagation*.

distributed application
An application for which the component application programs are distributed between two or more interconnected processors.

distributed identity
User identity information that originates from a remote system. The distributed identity is created in one system and is passed to one or more other systems over a network. See also *distinguished name* and *realm name*.

distributed processing
The processing of different parts of the same application in different systems, on one or more processors.

distributed program link (DPL)
A link that enables an application program running on one CICS system to link to another application program running in another CICS system.

DLC See *data link control*.

DLL See *dynamic link library*.

domain
In the Internet, a part of a naming hierarchy in which the domain name consists of a sequence of names (labels) separated by periods (dots).

domain name
In TCP/IP, a name of a host system in a network.

domain name server
In TCP/IP, a server program that supplies name-to-address translation by mapping domain names to IP addresses. Synonymous with name server.

dotted decimal notation
The syntactical representation for a 32-bit integer that consists of four 8-bit numbers written in base 10 with periods (dots) separating them. It is used to represent IP addresses.

double-byte character set (DBCS)
A set of characters in which each character is represented by 2 bytes. Languages such as Japanese, Chinese and Korean, which contain more symbols than can be represented by 256 code points, require double-byte character sets. Because each character requires 2 bytes, the typing, display, and printing of DBCS characters requires hardware and programs that support DBCS. Contrast with *single-byte character set.*

DPL See *distributed program link*.

dynamic link library (DLL)
A collection of runtime routines made available to applications as required.

dynamic server selection (DSS)
The mapping of a logical CICS server name to an actual CICS server name at run time.

E

EBCDIC
See *extended binary-coded decimal interchange code*.

ECI See *external call interface*.

EJB See *Enterprise JavaBeans*.

emulation program
A program that allows a host system to communicate with a workstation in the same way as it would with the emulated terminal.

emulator
A program that causes a computer to act as a workstation attached to another system.

encryption
The process of transforming data into an unintelligible form in such a way that the original data can be obtained only by using a decryption process.

enterprise bean
A Java component that can be combined with other resources to create JEE applications. There are three types of enterprise beans: entity beans, session beans, and message-driven beans.

Enterprise Information System (EIS)
The applications that comprise an enterprise's existing system for handling

company-wide information. An enterprise information system offers a well-defined set of services that are exposed as local or remote interfaces or both.

Enterprise JavaBeans (EJB)
A component architecture defined by Sun Microsystems for the development and deployment of object-oriented, distributed, enterprise-level applications (JEE).

environment variable
A variable that specifies the operating environment for a process. For example, environment variables can describe the home directory, the command search path, the terminal in use, and the current time zone.

EPI See *external presentation interface*.

ESI See *external security interface*.

Ethernet
A local area network that allows multiple stations to access the transmission medium at will without prior coordination, avoids contention by using carrier sense and deference, and resolves contention by using collision detection and transmission. Ethernet uses carrier sense multiple access with collision detection (CSMA/CD).

EXCI See *external CICS interface*.

extended binary-coded decimal interchange code (EBCDIC)
A coded character set of 256 8-bit characters developed for the representation of textual data.

extended logical unit of work (extended LUW)
A logical unit of work that is extended across successive ECI requests to the same CICS server.

external call interface (ECI)
A facility that allows a non CICS program to run a CICS program. Data is exchanged in a COMMAREA or a channel as for usual CICS interprogram communication.

external communications interface (EXCI)
An MVS application programming interface provided by CICS Transaction Server for z/OS that enables a non-CICS program to call a CICS program and to pass and receive data using a COMMAREA. The CICS application program is started as if linked-to by another CICS application program.

external presentation interface (EPI)
A facility that allows a non CICS program to appear to CICS as one or more standard 3270 terminals. 3270 data can be presented to the user by emulating a 3270 terminal or by using a graphical user interface.

external security interface (ESI)
A facility that enables client applications to verify and change passwords for user IDs on CICS servers.

External Security Manager (ESM)
A security manager that operates outside CICS. For example, RACF can be used as an external security manager with CICS Transaction Server.

F

firewall
A configuration of software that prevents unauthorized traffic between a trusted network and an untrusted network.

FMH See *function management header*.

fully-qualified APPLID
Used to identify CICS Transaction Gateway connections on the CICS server and tasks in a CICSplex. It is composed of an APPLID with an optional network qualifier. See also *APPLID* and *APPLID qualifier*.

function management header (FMH)
One or more headers, optionally present in the leading request units (RUs) of an RU chain, that allow one LU to (a) select a transaction program or device at the session partner and control the way in which the end-user data it sends is handled at the destination, (b) change the destination or the characteristics of the data during the session, and (c) transmit between session partners status or user information about the destination (for example, a program or device). Function management headers can be used with LU type 1, 4, and 6.2 protocols.

G

Gateway
A device or program used to connect two systems or networks.

Gateway classes
The Gateway classes provide APIs for ECI, EPI, and ESI that allow communication between Java client applications and the Gateway daemon.

Gateway daemon
A long-running Java process that listens for network requests from remote Client applications. It issues these requests to CICS servers using the CICS connectivity components. The Gateway daemon on z/OS processes ECI requests and on UNIX, Windows, and Linux platforms it process EPI and ESI requests as well. The Gateway daemon uses the GATEWAY section of ctg.ini for its configuration.

Gateway group
A set of Gateway daemons that share an APPLID qualifier, and where each Gateway daemon has a unique APPLID within the Gateway group.

Gateway token
A token that represents a specific Gateway daemon, when a connection is established successfully. Gateway tokens are used in the C language statistics and ECI V2 APIs.

global transaction
A recoverable unit of work performed by one or more resource managers in a distributed transaction processing environment and coordinated by an external transaction manager.

H

HA group
See *highly available Gateway group*.

highly available Gateway group (HA group)
A Gateway group that utilizes TCP/IP load balancing, and can be viewed

as a single logical Gateway daemon. A Gateway daemon instance in a HA group can recover indoubt XA transactions on behalf of another Gateway daemon within the HA group

host A computer that is connected to a network (such as the Internet or an SNA network) and provides an access point to that network. The host can be any system; it does not have to be a mainframe.

host address
An IP address that is used to identify a host on a network.

host ID
In TCP/IP, that part of the IP address that defines the host on the network. The length of the host ID depends on the type of network or network class (A, B, or C).

host name
In the Internet suite of protocols, the name given to a computer. Sometimes, host name is used to mean the fully qualified domain name; other times, it is used to mean the most specific subname of a fully qualified domain name. For example, if mycomputer.city.company.com is the fully qualified domain name, either of the following can be considered the host name: mycomputer.city.company.com, mycomputer.

hover help
Information that can be viewed by holding a mouse over an item such as an icon in the user interface.

HTTP See *Hypertext Transfer Protocol*.

HTTPS
See *Hypertext Transfer Protocol Secure*.

Hypertext Transfer Protocol (HTTP)
In the Internet suite of protocols, the protocol that is used to transfer and display hypertext and XML documents.

Hypertext Transfer Protocol Secure (HTTPS)
A TCP/IP protocol that is used by World Wide Web servers and Web browsers to transfer and display hypermedia documents securely across the Internet.

I

ID data
An ID data structure holds an individual result from a statistical API function.

identity propagation
The concept of preserving a user's security identity information (the distributed identity) independent of where the identity information has been created, for use during authorization and for auditing purposes. The distributed identity is carried with a request from the distributed client application to the CICS server, and is incorporated in the access control of the server as part of the authorization process, for example, using RACF. CICS Transaction Gateway flows the distributed identity to CICS. See also *distributed identity*.

identity propagation login module
A code component that provides support for identity propagation. The identity propagation login module is included with the CICS Transaction

Gateway ECI resource adapter (cicseci.rar), conforms to the JAAS specification and is contained in a single Java class within the resource adapter. See also *identity propagation*.

iKeyman
A tool for maintaining digital certificates for JSSE.

in doubt
The state of a transaction that has completed the prepare phase of the two-phase commit process and is waiting to be completed.

in flight
The state of a transaction that has not yet completed the prepare phase of the two-phase commit process.

independent logical unit
A logical unit (LU) that can both send and receive a BIND, and which supports single, parallel, and multiple sessions. See *BIND*.

<install_path>
This term is used in file paths to represent the directory where you installed the product. For more information, see File path terminology.

Internet Architecture Board
The technical body that oversees the development of the internet suite of protocols known as TCP/IP.

Internet Protocol (IP)
In TCP/IP, a protocol that routes data from its source to its destination in an Internet environment.

interoperability
The capability to communicate, run programs, or transfer data among various functional units in a way that requires the user to have little or no knowledge of the unique characteristics of those units.

IP Internet Protocol.

IPIC See *IP interconnectivity*.

IP address
A unique address for a device or logical unit on a network that uses the IP standard.

IP interconnectivity (IPIC)
The IPIC protocol enables Distributed Program Link (DPL) access from a non-CICS program to a CICS program over TCP/IP, using the External Call Interface (ECI). IPIC passes and receives data using COMMAREAs, or containers.

J

JEE (formerly J2EE)
See *Java 2 Platform Enterprise Edition*

JEE Connector architecture (JCA)
A standard architecture for connecting the JEE platform to heterogeneous enterprise information systems (EIS).

Java An object-oriented programming language for portable interpretive code that supports interaction among remote objects.

Java 2 Platform Enterprise Edition (JEE)
An environment for developing and deploying enterprise applications,

defined by Sun Microsystems Inc. The JEE platform consists of a set of services, application programming interfaces (APIs), and protocols that allow multi-tiered, Web-based applications to be developed.

JavaBeans
As defined for Java by Sun Microsystems, a portable, platform-independent, reusable component model.

Java Client application
The Java client application is a user application written in Java, including servlets and enterprise beans, that uses the Gateway classes.

Java Development Kit (JDK)
The name of the software development kit that Sun Microsystems provided for the Java platform, up to and including v 1.1.x. Sometimes used erroneously to mean the Java platform or as a generic term for any software developer kits for Java.

JavaGateway
The URL of the CICS Transaction Gateway with which the Java Client application communicates. The JavaGateway takes the form `protocol://address:port`. These protocols are supported: `tcp://`, `ssl://`, and `local:`. CICS Transaction Gateway runs with the default port value of 2006. This parameter is not relevant if you are using the protocol `local:`. For example, you might specify a JavaGateway of `tcp://ctg.business.com:2006`. If you specify the protocol as `local:` you will connect directly to the CICS server, bypassing any CICS Transaction Gateway servers.

Java Native Interface (JNI)
A programming interface that allows Java code running in a Java virtual machine to work with functions that are written in other programming languages.

Java Runtime Environment (JRE)
A subset of the Java Software Development Kit (SDK) that supports the execution, but not the development, of Java applications. The JRE comprises the Java Virtual Machine (JVM), the core classes, and supporting files.

Java Secure Socket Extension (JSSE)
A Java package that enables secure Internet communications. It implements a Java version of the Secure Sockets Layer (SSL) and Transport Layer Security (TSL) protocols and supports data encryption, server authentication, message integrity, and optionally client authentication.

Java virtual machine (JVM)
A software implementation of a processor that runs compiled Java code (applets and applications).

JDK See *Java development kit*.

JCA See *JEE Connector Architecture* .

JNI See *Java Native Interface*.

JRE See *Java Runtime Environment*

JSSE See *Java Secure Socket Extension*.

JVM See *Java Virtual Machine*.

K

keyboard mapping
A list that establishes a correspondence between keys on the keyboard and characters displayed on a display screen, or action taken by a program, when that key is pressed.

Keystore
In the JSSE protocol, a file that contains public keys, private keys, trusted roots, and certificates.

L

local mode
Local mode describes the use of the CICS Transaction Gateway *local* protocol. The Gateway daemon is not used in local mode.

local transaction
A recoverable unit of work managed by a resource manager and not coordinated by an external transaction manager.

logical CICS server
An alias that can be passed on an ECI request when running in remote mode to CICS Transaction Gateway for z/OS. The alias name is mapped to an actual CICS server name by a dynamic server selection (DSS) mechanism.

logical end of day
The local time of day on the 24-hour clock to which a Gateway daemon aligns statistics intervals. If the statistics interval is 24 hours, this is the local time at which interval statistics will be reset and, on z/OS, optionally recorded to SMF. This time is set using the **`stateod`** parameter in the configuration file (ctg.ini).

logical unit (LU)
In SNA, a port through which an end user accesses the SNA network to communicate with another end user and through which the end user accesses the functions provided by system services control points (SSCP). An LU can support at least two sessions, one with an SSCP and one with another LU, and might be capable of supporting many sessions with other logical units. See also *network addressable unit, primary logical unit, secondary logical unit*.

logical unit 6.2 (LU 6.2)
A type of logical unit that supports general communications between programs in a distributed processing environment.

The LU type that supports sessions between two applications using APPC.

logical unit of work (LUW)
The processing that a program performs between synchronization points

LU See *logical unit*.

LU 6.2 See *logical unit 6.2*.

LU-LU session
In SNA, a session between two logical units (LUs) in an SNA network. It provides communication between two end users, or between an end user and an LU services component.

LU-LU session type 6.2
In SNA, a type of session for communication between peer systems. Synonymous with APPC protocol.

LUW See *logical unit of work*.

M

managed mode
Describes an environment in which connections are obtained from connection factories that the JEE server has set up. Such connections are owned by the JEE server.

media access control (MAC) sublayer
One of two sublayers of the ISO Open Systems Interconnection data link layer proposed for local area networks by the IEEE Project 802 Committee on Local Area Networks and the European Computer Manufacturers Association (ECMA). It provides functions that depend on the topology of the network and uses services of the physical layer to provide services to the logical link control (LLC) sublayer. The OSI data link layer corresponds to the SNA data link control layer.

method
In object-oriented programming, an operation that an object can perform. An object can have many methods.

mode In SNA, a set of parameters that defines the characteristics of a session between two LUs.

N

name server
In TCP/IP, synonym for Domain Name Server. In Internet communications, a host that translates symbolic names assigned to networks and hosts into IP addresses.

NAU See *network addressable unit*.

network address
In SNA, an address, consisting of subarea and element fields, that identifies a link, link station, or network addressable unit (NAU). Subarea nodes use network addresses; peripheral nodes use local addresses. The boundary function in the subarea node to which a peripheral node is attached transforms local addresses to network addresses and vice versa. See also *network name*.

network addressable unit (NAU)
In SNA, a logical unit, a physical unit, or a system services control point. The NAU is the origin or the destination of information transmitted by the path control network. See also *logical unit*, *network address*, *network name*.

network name
In SNA, the symbolic identifier by which end users refer to a network addressable unit (NAU), link station, or link. See also *network address*.

node type
In SNA, a designation of a node according to the protocols it supports and the network addressable units (NAUs) it can contain. Four types are defined: 1, 2, 4, and 5. Type 1 and type 2 nodes are peripheral nodes; type 4 and type 5 nodes are subarea nodes.

nonextended logical unit of work
See *SYNCONRETURN*.

nonmanaged mode
An environment in which the application is responsible for generating and

configuring connection factories. The JEE server does not own or know about these connection factories and therefore provides no Quality of Service facilities.

O

object In object-oriented programming, a concrete realization of a class that consists of data and the operations associated with that data.

object-oriented (OO)
Describing a computer system or programming language that supports objects.

one-phase commit
A protocol with a single commit phase, that is used for the coordination of changes to recoverable resources when a single resource manager is involved.

OO See *object-oriented*.

P

pacing
A technique by which a receiving station controls the rate of transmission of a sending station to prevent overrun.

parallel session
In SNA, two or more concurrently active sessions between the same two LUs using different pairs of network addresses. Each session can have independent session parameters.

PING In Internet communications, a program used in TCP/IP networks to test the ability to reach destinations by sending the destinations an Internet Control Message Protocol (ICMP) echo request and waiting for a reply.

partner logical unit (PLU)
In SNA, the remote participant in a session.

partner transaction program
The transaction program engaged in an APPC conversation with a local transaction program.

password phrase
A character string, between 9 and 100 characters in length, that is used for authentication when a user signs on to CICS. Because a password phrase can provide an exponentially greater number of possible combinations of characters than a standard 8 character password, the use of password phrases can enhance system security. Password phrases are verified by the External Security Manager (ESM), and can contain alphanumeric characters, and any of the other non alphanumeric characters that are supported by the ESM. See also *External Security Manager (ESM)*.

PLU See *primary logical unit* and *partner logical unit*.

policy-based dynamic server selection (DSS)
A selection mechanism that CICS transaction Gateway uses when deciding which CICS servers will receive workload. Policy-based DSS ensures that requests are sent to targeted groups of CICS servers, and that CICS servers within the groups are selected for workload using a specified algorithm (round robin or failover).

port An endpoint for communication between devices, generally referring to a logical connection. A 16-bit number identifying a particular Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) resource within a given TCP/IP node.

port sharing
A way of load balancing TCP/IP connections across a group of servers running in the same z/OS image.

prepare phase
The first phase of a XA process in which all participants are requested to confirm readiness to commit.

presentation logic
The part of a distributed application that is concerned with the user interface of the application. Compare with *business logic*.

primary logical unit (PLU)
In SNA, the logical unit that contains the primary half-session for a particular logical unit-to-logical unit (LU-to-LU) session. See also *secondary logical unit*.

<product_data_path>
This term represents the directory used by the Windows CICS Transaction Gateway for common application data. For more information, see File path terminology.

protocol boundary
The signals and rules governing interactions between two components within a node.

Q

Query strings
Query strings are used in the statistical data API. A query string is an input parameter, specifying the statistical data to be retrieved.

R

RACF See *Resource Access Control Facility*.

realm A named collection of users and groups that can be used in a specific security context. See also *distinguished name* and *identity propagation*.

Recoverable resource management services (RRMS)
The registration services, context services, and resource recovery services provided by the z/OS sync point manager that enable consistent changes to be made to multiple protected resources.

Resource Access Control Facility (RACF)
An IBM licensed program that provides access control by identifying users to the system; verifying users of the system; authorizing access to protected resources; logging detected unauthorized attempts to enter the system; and logging detected accesses to protected resources.

region In workload management on CICS Transaction Gateway for Windows, an instance of a CICS server.

remote mode
Remote mode describes the use of one of the supported CICS Transaction Gateway network protocols to connect to the Gateway daemon.

remote procedure call (RPC)
A protocol that allows a program on a client computer to run a program on a server.

Request monitoring exits
Exits that provide information about individual requests as they are processed by the CICS Transaction Gateway.

request unit (RU)
In SNA, a message unit that contains control information such as a request code, or function management (FM) headers, end-user data, or both.

request/response unit
A generic term for a request unit or a response unit. See also *request unit* and *response unit*.

response file
A file that contains predefined values that is used instead of someone having to enter those values one at a time. See also *CID methodology*.

response unit (RU)
A message unit that acknowledges a request unit; it can contain prefix information received in a request unit.

Resource adapter
A system-level software driver that is used by an EJB container or an application client to connect to an enterprise information system (EIS). A resource adapter plugs in to a container; the application components deployed on the container then use the client API (exposed by adapter) or tool-generated, high-level abstractions to access the underlying EIS.

resource group ID
A resource group ID is a logical grouping of resources, grouped for statistical purposes. A resource group ID is associated with a number of resource group statistics, each identified by a statistic ID.

resource ID
A resource ID refers to a specific resource. Information about the resource is included in resource-specific statistics. Each statistic is identified by a statistic ID.

resource manager
The participant in a transaction responsible for controlling access to recoverable resources. In terms of the CICS resource adapters this is represented by an instance of a ConnectionFactory.

Resource Recovery Services (RRS)
A z/OS facility that provides two-phase sync point support across participating resource managers.

Result set
A result set is a set of data calculated or recorded by a statistical API function.

Result set token
A result set token is a reference to the set of results returned by a statistical API function.

rollback
An operation in a transaction that reverses all the changes made during the unit of work. After the operation is complete, the unit of work is finished. Also known as a backout.

RU See *Request unit* and *Response unit*.

RPC See *remote procedure call*.

RRMS
See *Recoverable resource management services*.

RRS See *Resource Recovery Services*.

S

SBCS See *single-byte character set*.

secondary logical unit (SLU)
In SNA, the logical unit (LU) that contains the secondary half-session for a particular LU-LU session. Contrast with primary logical unit. See also *logical unit*.

Secure Sockets Layer (SSL)
A security protocol that provides communication privacy. SSL enables client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, and message forgery. SSL applies only to internet protocols, and is not applicable to SNA.

server name remapping
See *dynamic server selection*.

servlet
A Java program that runs on a Web server and extends the server's functionality by generating dynamic content in response to Web client requests. Servlets are commonly used to connect databases to the Web.

session limit
In SNA, the maximum number of concurrently active logical unit to logical unit (LU-to-LU) sessions that a particular logical unit (LU) can support.

silent installation
Installation that does not display messages or windows during its progress. Silent installation is not a synonym of "unattended installation", although it is often improperly used as such.

single-byte character set (SBCS)
A character set in which each character is represented by 1 byte. Contrast with double-byte character set.

sign-on capable terminal
A sign-on capable terminal allows sign-on transactions that are either supplied with CICS (CESN) or written by the user, to be run. Contrast with sign-on incapable terminal.

SIT See *system initialization table*.

SLU See *secondary logical unit*.

SMF The z/OS System Management Facility (SMF) collects and records system and job-related information that your z/OS installation can use for reporting, billing, analysis, profiling, and maintaining system security. CICS TG for 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.

statistic data
A statistic data structure holds individual statistical result returned after calling a statistical API function.

statistic group
A generic term for a collection of statistic IDs.

statistic ID
A label referring to a specific statistic. A statistic ID is used to retrieve specific statistical data, and always has a direct relationship with a statistic group.

standard error
In many workstation-based operating systems, the output stream to which error messages or diagnostic messages are sent.

subnet
An interconnected, but independent segment of a network that is identified by its Internet Protocol (IP) address.

subnet address
In Internet communications, an extension to the basic IP addressing scheme where a portion of the host address is interpreted as the local network address.

sync point
Synchronization point. During transaction processing, a reference point to which protected resources can be restored if a failure occurs.

SYNCONRETURN
A request where the CICS server takes a sync point on successful completion of the server program. Changes to recoverable resources made by the server program are committed or rolled-back independently of changes to recoverable resources made by the client program issuing the ECI request, or changes made by the server in any subsequent ECI request. Also referred to as a *nonextended logical unit of work*.

system initialization table (SIT)
A table containing parameters used to start a CICS control region.

System Management Command
An administrative request received by a Gateway daemon (or Gateway daemon address space on z/OS) from the **`ctgadmin`** command (on UNIX, Linux, or Windows) or the z/OS console. The request might be made to

retrieve information about the Gateway daemon, or to alter some aspect of Gateway daemon behavior. Typically, a **`ctgadmin`** command in the form **`ctgadmin`** <command string> is entered by an operator using the command line interface, or a modify command in the form `/F <job name>,APPL=<command string>` is entered by an operator on the z/OS console.

System Management Interface Tool (SMIT)
An interface tool of the AIX operating system for installing, maintaining, configuring, and diagnosing tasks.

Systems Network Architecture (SNA)
An architecture that describes the logical structure, formats, protocols, and operational sequences for transmitting information units through the networks and also the operational sequences for controlling the configuration and operation of networks.

System SSL
An implementation of SSL, no longer supported by CICS Transaction Gateway on z/OS.

T

TCP/IP
See *Transmission Control Protocol/Internet Protocol*.

TCP/IP load balancing
The ability to distribute TCP/IP connections across target servers.

terminal emulation
The capability of a personal computer to operate as if it were a particular type of terminal linked to a processing unit and to access data. See also *emulator, emulation program*.

thread A stream of computer instructions that is in control of a process. In some operating systems, a thread is the smallest unit of operation in a process. Several threads can run concurrently, performing different jobs.

timeout
A time interval that is allotted for an event to occur or complete before operation is interrupted.

TLS See *Transport Layer Security*.

token-ring network
A local area network that connects devices in a ring topology and allows unidirectional data transmission between devices by a token-passing procedure. A device must receive a token before it can transmit data.

trace A record of the processing of a computer program. It exhibits the sequences in which the instructions were processed.

transaction manager
A software unit that coordinates the activities of resource managers by managing global transactions and coordinating the decision to commit them or roll them back.

transaction program
A program that uses the Advanced Program-to-Program Communications (APPC) application programming interface (API) to communicate with a partner application program on a remote system.

Transmission Control Protocol/Internet Protocol (TCP/IP)
An industry-standard, nonproprietary set of communications protocols that provide reliable end-to-end connections between applications over interconnected networks of different types.

Transport Layer Security (TLS)
A security protocol that provides communication privacy. TLS enables client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, and message forgery. TLS applies only to internet protocols, and is not applicable to SNA. TLS is also known as SSL 3.1.

Two-phase commit
A protocol with both a prepare and a commit phase, that is used for the coordination of changes to recoverable resources when more than one resource manager is used by a single transaction.

type 2.0 node
A node that attaches to a subarea network as a peripheral node and provides a range of end-user services but no intermediate routing services.

type 2.1 node
An SNA node that can be configured as an endpoint or intermediate routing node in a network, or as a peripheral node attached to a subarea network.

U

unattended installation
Unattended installation is installation performed without user interaction during its progress, or, with no user present at all, except for the initial launch of the process. -

Uniform Resource Locator (URL)
A sequence of characters that represent information resources on a computer or in a network such as the Internet. This sequence of characters includes (a) the abbreviated name of the protocol used to access the information resource and (b) the information used by the protocol to locate the information resource.

unit of recovery (UR)
A defined package of work to be performed by the RRS.

unit of work (UOW)
A recoverable sequence of operations performed by an application between two points of consistency. A unit of work begins when a transaction starts or at a user-requested sync point. It ends either at a user-requested sync point or at the end of a transaction.

UOW See *unit of work*.

UR See *unit of recovery*.

URL See *Uniform Resource Locator*.

user registry
The location where the distinguished name of a user is defined and authenticated. See also *distinguished name*.

user session
Any APPC session other than a SNASVCMG session.

V

verb A reserved word that expresses an action to be taken by an application programming interface (API), a compiler, or an object program.

In SNA, the general name for a transaction program's request for communication services.

version string
A character string containing version information about the statistical data API.

W

WAN See *wide area network*.

Web browser
A software program that sends requests to a Web server and displays the information that the server returns.

Web server
A software program that responds to information requests generated by Web browsers.

wide area network (WAN)
A network that provides communication services to a geographic area larger than that served by a local area network or a metropolitan area network, and that can use or provide public communication facilities.

Wrapping trace
On Windows, UNIX, and Linux, a configuration in which the **Maximum Client wrap size** setting is greater than 0. The total size of Client daemon binary trace files is limited to the value specified in the **Maximum Client wrap size** setting. With standard I/O tracing, two files, called `cicscli.bin` and `cicscli.wrp`, are used; each can be up to half the size of the **Maximum Client wrap size**.

X

XA request
Any request sent or received by the CICS Transaction Gateway in support of an XA transaction. These requests include the XA commands commit, complete, end, forget, prepare, recover, rollback, and start.

XA transaction
A global transaction that adheres to the X/Open standard for distributed transaction processing (DTP.)

Index

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte 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

The following paragraph does not apply in the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who want to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM United Kingdom

Laboratories, MP151, Hursley Park, Winchester, Hampshire, England, SO21 2JN. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines 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.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Readers' Comments — We'd Like to Hear from You

CICS Transaction Gateway
Version 9 Release 0
z/OS Administration

Publication No. SC34-2832-02

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

SC34-2832-02

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 (MP095)
Hursley Park
Winchester
Hampshire
United Kingdom
SO21 2JN

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

SC34-2832-02